AFOSR-TR- 87-0802

THE BDM CORPORATION

7915 Jones Branch Drive, McLean, Virginia 22102-3396 • (703) 848-5000 • Telex: 901103 BDM MCLN

Optics and Symbolic Computing Semi-Annual Technical Report

PREPARED FOR AIR FORCE OFFICE OF SCIENTIFIC RESEARCH (AFOSR)
AND DEFENSE ADVANCED RESEARCH PROJECTS AGENCY (DARPA)

MARCH 1987

BDM/MCL-87-0097-TR

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release Distribution Unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S) AFOSR-TR- 87-0802
6a. NAME OF PERFORMING ORGANIZATION BDM CORP — 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION AFOSR
6c. ADDRESS (City, State and ZIP Code) 7915 Jones Branch Drive McLean, VA 22102-3396	7b. ADDRESS (City, State and ZIP Code) Bldg 410 Bolling AFB, DC 20332-6448
8a. NAME OF FUNDING/SPONSORING ORGANIZATION AFOSR — 8b. OFFICE SYMBOL (If applicable) NE	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F49620-86-C-0030
8c. ADDRESS (City, State and ZIP Code) Bldg 410 Bolling AFB, DC 20332-6448	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO.: 61102F — PROJECT NO.: 4952 — TASK NO.: 02 DARPA — WORK UNIT NO.:
11. TITLE (Include Security Classification) ~~Application of Optical Computing to problems with Symbolic Computations~~ See Cover	
12. PERSONAL AUTHOR(S) Dr Brian Kushner	
13a. TYPE OF REPORT Semi-Annual — 13b. TIME COVERED FROM Feb 86 TO Aug 86	14. DATE OF REPORT (Yr., Mo., Day) 87 March — 15. PAGE COUNT 286
16. SUPPLEMENTARY NOTATION	
17. COSATI CODES FIELD / GROUP / SUB. GR.	18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

There exists a set of problems that conventional computer science approaches find intractable or phohibitively expensive. This set includes image, natural language and speech underestanding, along with expert reasoning. Interestingly enough, thsee proble are easily solved by humans so a solution must exist. The computer science community has taken up this challenge and the field of Artificial Intelligence (AI) developed to address these problems.

Unfortunately, the solution strategies developed by the AI researchers so far posses several shortcomings. In particular, to implement the solutions for meaningful applications requires the development of computers far different than the digital electronic workhorses of the past. It is not clear whether the presently available electronic computer technology is even capable of building machines that can solve these problems. Therefore, in this report we identified ways how an alternative technology, optical computing, may address the critical shortcomings of electronic approaches.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ OTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL DR C.L GILES	22b. TELEPHONE NUMBER (Include Area Code) (202) 767-4931 — 22c. OFFICE SYMBOL NE

DD FORM 1473, 83 APR · EDITION OF 1 JAN 73 IS OBSOLETE.

SECURITY CLASSIFICATION OF THIS PA

THE BDM CORPORATION

7915 Jones Branch Drive, McLean, Virginia 22102-3396 • (703) 848-5000 • Telex: 901103 BDM MCLN

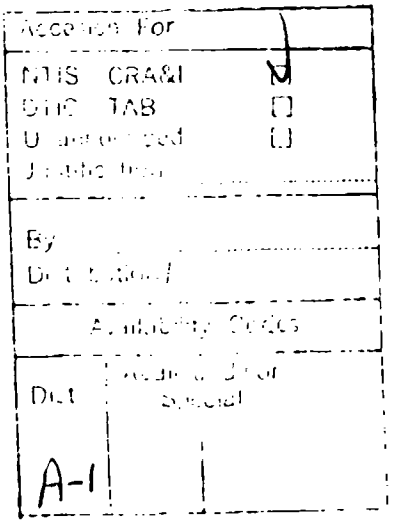
OPTICS AND SYMBOLIC COMPUTING
SEMI-ANNUAL TECHNICAL REPORT
March 1987

BDM/MCL-87-0097-TR

Sponsored by
Advanced Research Projects Agency (DOD)
ARPA Order No. 4952
Monitored by AFOSR Under Contract #F49620-86-C-0030

Prepared for Air Force Office of Scientific Research (AFOSR) and Defense Advanced Research Projects Agency (DARPA).

TABLE OF CONTENTS



The title on the front cover is correct.
Per Ms. Debbie Tyrell, AFOSR/XOTD

LIST OF FIGURES

LIST OF FIGURES (CONTINUED)

LIST OF FIGURES (CONTINUED)

LIST OF FIGURES (CONTINUED)

LIST OF TABLES

SUMMARY

SUMMARY

There exists a set of problems that conventional computer science approaches find intractable or prohibitively expensive. This set includes image, natural language and speech understanding, along with expert reasoning. Interestingly enough, these problems are easily solved by humans so a solution must exist. The computer science community has taken up this challenge and the field of Artificial Intelligence (AI) developed to address these problems. Unfortunately, the solution strategies developed by the AI researches so far possess several shortcomings. In particular, to implement the solutions for meaningful applications requires the development of computers far different than the digital electronic workhorses of the past. It is not clear whether the presently available electronic computer technology is even capable of building machines that can solve these problems. Therefore, in this report we identified ways how an alternative technology, optical computing, may address the critical shortcomings of electronic approaches.

In pursuit of this goal, the first section of this report, a book chapter and a paper entitled "Optics and Symbolic Computing", delineates the focus of a few potentially fruitful research efforts for optical computing scientists. The goal of this approach is to review traditional AI problems and methods and identify critical points where the use of optical techniques may allow higher computational throughputs. The chapter begins by defining symbolic computing with regard to the characteristics and representations of knowledge. Moreover, this section highlights the typical operations found in a symbolic computing system and contrasts their attributes with those of conventional numeric computing. Next the authors review the strategies for solving the traditional AI problems and identify the computational bottlenecks in proposed systems. In particular, the potential for optics to address critical bottlenecks is identified. The chapter concludes by introducing sequential and parallel symbolic computing architectures and their possible optical implementations. The optical architectures proposed range from replacement of electrical connections

with optical connections to all optical machines. Avenues for future research that will provide the most benefits are the following: high-speed global interconnects, reconfigurable interconnects, high fan-out optical elements, and multiport components for parallel processing.

The AI problem most suited for the immediate impact of optics is image understanding. Therefore, the next section explores the algorithms and implementations of image understanding by computers. The significance of image understanding to military systems and advanced manufacturing environments is growing because of the need for remote and intelligent sensors. Hence the report cites many image understanding algorithms that may have parallel implementations suitable for optical systems. The role for optics is identified as a low-level, front-end preprocessor for a hybrid digital-optical system. The impact of efficient and rich low-level operations on higher level symbolic processing is emphasized as the main reason why a modular optical preprocessing system with high-level control is needed. The interface and feedback control between the high and low level operations are identified as the critical components for the construction of a workable image understanding engine. In this vein, relaxation is proposed to bridge the gap between all processing stages. It is compatible with the modular and highly parallel processors possible with optics and may be useful in reconstruction of a meaningful representation of a scene.

Another candidate problem for optics based symbolic computers is in the area of adaptive knowledge-based systems. In a paper presented at FJCC entitled "An Organizational Framework for Comparing Adaptive Artificial Intelligence Systems", the inherent limitations of traditional expert systems are contrasted with the advantages of adaptive systems. Various adaptive systems are compared with respect to the following features: representation; storage and recall mechanisms; control strategy; adaptation of storage and control structure. From this analysis, the design of an ideal adaptive system is outlined. Since expert systems of the future will probably resemble the adaptive systems of today, optical machines for knowledge-based systems should incorporate some of the features presented in this paper.

A critical feature of knowledge-based AI systems is the need for rapid recall of data from incomplete, contradictory and noisy information. In this regard, optical associative memories that possess these capabilities may play a key role in the development of robust AI systems. Therefore, in this section, an associative memory is proposed that has a large storage capacity and can focus its attention on specific information in response to external influences. Attention is crucial in constraining the search procedure. Relational data base machines are described and the need for a processor that can associate data is established. The associative memory presented in this section can be cascaded to form a heteroassociative memory or with the addition of logic units perform many functions required in relational algebra machines.

In addition to optical relational algebra machines, optical matrix algebra processors may have a tremendous import to the realization of optical symbolic computers. Specifically matrix algebra is useful in low level image processing and may enable the development of inference engines and directed graph processors. In light of their significance, optical matrix algebra processor architectures were classified according to the degree of parallelism employed and the type of interconnections.

In conclusion, optics may be able to enhance the performance of computers applied to AI problems. Several candidate areas where the benefits of optics would increase the capabilities of proposed architectural implementations were identified. Where appropriate, specific optical architectures were proposed. This semi-annual technical report is a compilation of technical reports, briefings, and papers delivered under the terms of the contract. Future research will focus on the role of optics in ubiquitous symbolic operations like compare-and-exchange and pattern matching. In addition, new computational models that may be better suited for describing symbolic computation will be considered.

OPTICS AND SYMBOLIC COMPUTING

CHAPTER I
INTRODUCTION

The incorporation of intelligence into computational systems is rapidly gaining momentum with both the computer science community and with a large segment of computer system users. The field has been traditionally labeled "artificial intelligence," or "AI" for short. It is difficult to exactly specify what is meant by AI, however, since intelligence is a relative merit which cannot be precisely quantified or defined. Some aspects of intelligence have been in computers from the beginning, even though AI conjures up an image of very advanced computer science. Afterall, memory capabilities are usually associated with intelligence and even the earliest computers incorporated some form of memory. A working understanding of what is meant by AI might succinctly be stated as imparting to computers those attributes which we associate with the human thought processes that are notably different from the way in which conventional computers operate.

An important characteristic of AI systems is their incorporation and utilization of knowledge in each computation. Computer scientists have been struggling for the last two decades to determine how best to represent knowledge in computing machines. Numerous techniques have evolved for creating, manipulating, and storing collections of symbolic structures. These structures, or knowledge elements, can be used to represent objects, events, knowledge about how to do things, and knowledge related to what is known (or meta-knowledge). Collectively, this set of symbolic structures is referred to as the knowledge base of the AI system.

Why does optical processing look attractive for performing symbolic manipulations or computing? This chapter is designed to answer this question in some detail, but on the surface one could readily cite computational throughput and operation compatibility. The need for computational throughput can be appreciated by comparing the throughput performance of numeric computers against that of dedicated LISP machines (LISP, or LISt Processing, being the symbolic computing language of choice in the US AI community). Figure 1 is representative of system performances, depicting a

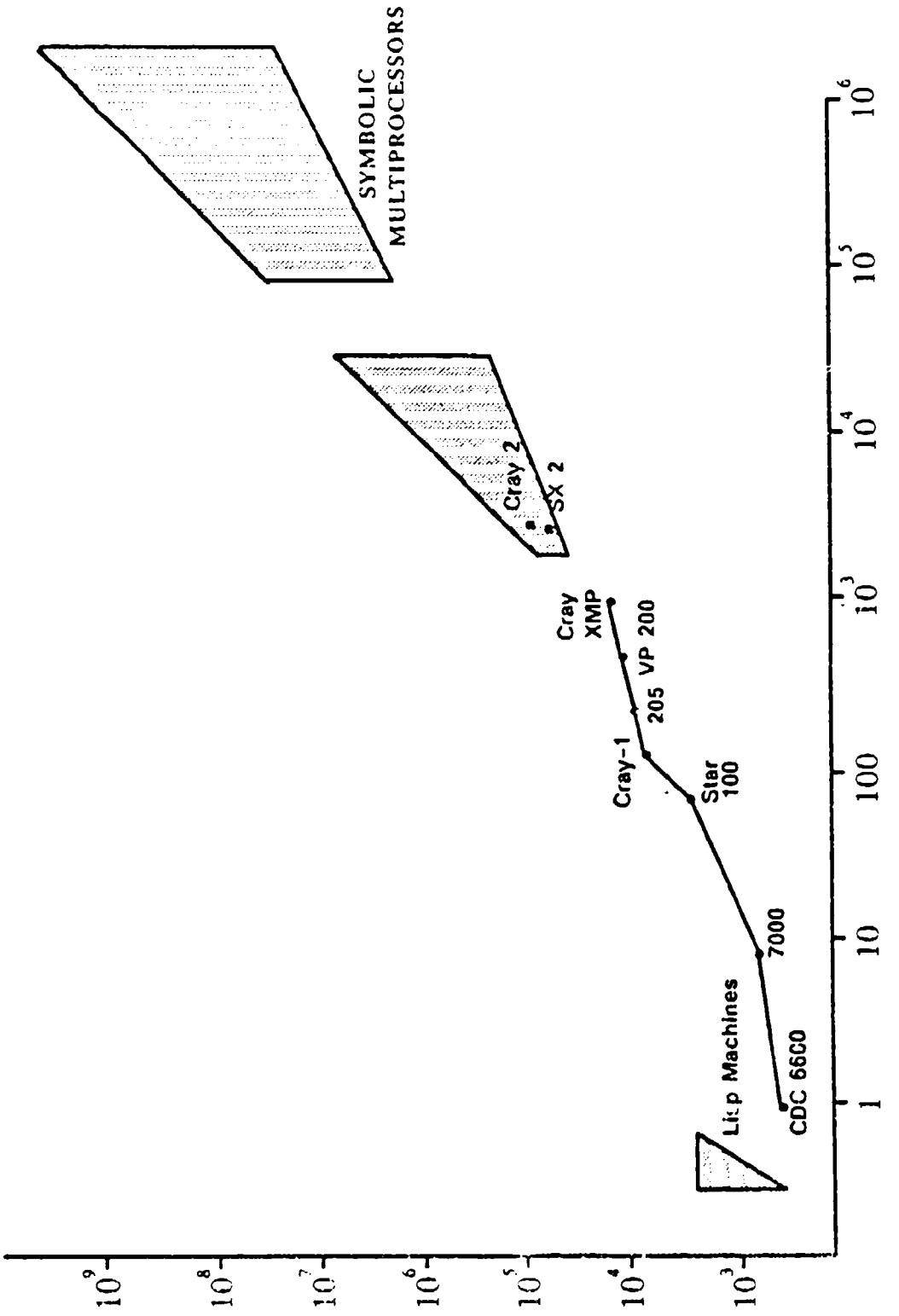


Figure 1. Trends in Numeric and Symbolic Computation Technology

(Source : IEEE)

typical measure of AI computing power on the vertical axis and operational speed on the horizontal axis.[1,2] LIPS, or Logical Inferences Per Second, is used here as a measure of intelligence, since no functional equivalent of an "IQ test" currently exists for AI systems. The figure illustrates the slow speed of these LISP machines, relative to current generation "supercomputers" and next generation multiprocessors. The need to drastically increase the processing rate of AI systems, coupled with the limitations of current uniprocessor architectures, has resulted in a major research impetus to explore parallelism to enhance the speed of these machines and render them far more useful as modern computing systems. This utilization of parallelism indicates a potential synergism between symbolic and optical processing.

Interestingly enough, the numeric "supercomputers" execute AI functions at a greater rate than dedicated AI machines. This may imply that raw speed is an important component in overcoming existing AI computational bottlenecks. It also may indicate that architectures designed to improve numeric throughput will be useful in symbolic computation, and vice versa. The issue of mapping the computational structure onto desired functionality is currently a global issue of concern, and crosses all boundaries of computer science and mathematics.

The second factor potentially linking optical and symbolic processing, that of operation compatibility, derives from the need to perform correlation, searching, and matching types of operations on symbolic data. Many of these operations do not require high computational accuracy. The application of optics to symbolic computing will likely avoid what has traditionally been the "Achilles heel" of optical computing - the difficulty in achieving more than a few bits of accuracy. In addition, the dependence of symbolic computing on correlation functions (and their isomorphs) may provide an excellent opportunity to enhance symbolic computing performance with the use of optical correlators.

The next section will describe the fundamental attributes of symbolic computing, and will compare it with techniques commonly utilized in numeric computing. Following this introductory discussion, we will present a

description of the most important functional capabilities that are based on symbolic computing. This will lead to a discussion of problem areas that are likely to be faced in achieving these capabilities. These sections do not address optical symbolic computing, but the reader will likely find the information contained therein to be important to gaining an understanding of the synergism between optical and symbolic forms of computing. In fact, these sections are intended as an introduction to the subject of artificial intelligence, and, because of the breadth of the topic, we have had to limit our discussions to only the major aspects. Section IV will then lay the framework for symbolic computing with optics. Fundamental architectural concepts will be described with an emphasis on how they differ from the more conventional computer architectures. This will be followed by the authors' concepts of how optics could enhance the performance of symbolic processors.

In trying to cover such a broad topic, we have attempted to provide an introduction to each of the main disciplines within symbolic computing. However, in many cases we have had to sacrifice the details of a particular topic in order to provide a balanced presentation. In other cases, the material is simply beyond the scope of this book. For additional information, the interested reader can consult literature such as The Handbook of Artificial Intelligence,[6] a three volume treatise which provides an excellent overview of AI technology.

CHAPTER II
WHAT IS SYMBOLIC COMPUTING?

In order to realize computers that are more capable of simulating human thought processes than is possible with today's numeric computers, the bit patterns within the computers must be made to represent arbitrary symbols in addition to arithmetic ones. For example, the computer should be able to provide an answer to the question "What path should I take to reach my destination (given all I know about my environment and my capabilities)?" as well as to an arithmetic question such as "What is the sum of 2 plus 4?" Humans routinely make routing decisions, but numeric computation was not developed to handle such problems. First of all, encoding the question itself into the computer offers a formidable problem. Second, if the destination and alternative path information is provided by a visual scene (i.e., visual navigation), the resulting object recognition and image understanding tasks can be immeasurably enhanced by symbolic manipulation. Finally, the decision process will likely draw on one's knowledge of the world, and this also relies on symbolic constructs.

A. KNOWLEDGE CHARACTERISTICS

A system that we would describe as knowledgeable has three main attributes: capability of acquiring additional knowledge, ability to retrieve appropriate information from a knowledge base, and the power of reasoning with the retrieved information to solve problems. In defining knowledge, we recognize that several other interpretations are possible, each with an independent set of characteristics. We believe that our set of attributes comprise a convenient definition for knowledgeable systems and one which will be used to provide a format for the following discussion; however, it suffers a malady common to any attempt to neatly dissect and categorize a complex phenomena - that of an inability to ascertain the independence of the attributes. This will become evident in the discussion

of acquisition in which reasoning (the third attribute) is described as a method of acquisition (first attribute) - this is what we know as learning.

Acquisition of knowledge can occur via three different routes as shown in Figure 2. First of all, knowledge can be placed into the computer by a programmer working in conjunction with a knowledge engineer. In the case of expert systems (to be discussed in Section III.E), the knowledge engineer acquires or extracts knowledge from an expert, and passes it onto the programmer, who embeds it within the computer. This process is referred to as knowledge engineering. Secondly, in the near future we expect more and more knowledge to be automatically acquired via sensors. This will become an increasingly popular technique with the advent of speech recognition and vision systems (to be discussed in Sections III.B and III.C). Both programmed and sensed acquisition may be accomplished by just rote techniques, but most often the information is classified in some way to facilitate the retrieval process. Classification is a process of associating each input with related items to form classes which aid significantly in identifying relevant data during knowledge base searches. Classification is also commonly known as linking and lumping. If one knows, upon acquiring a given piece of information, that it will be associated with an entity already in the knowledge base, then a link is specified between the two, and if many entities are likely to be used together, they are lumped into a larger structure. The reader should have a good appreciation and understanding of classification following the discussions on retrieval, knowledge representation, and search.

The third category of acquisition is learned knowledge. The field of machine learning is still in its infancy, but three areas that have shown recent progress are parameter adjustment, discovery, and analogical reasoning (ref. 18). The variation of parameters and stimuli is a standard scientific technique for learning. Two important areas of application for AI are in variation of classification parameters for knowledge acquisition (changing of classes into which objects are placed), and in adjusting heuristic function parameters for improved problem solving (to be discussed in Section II.C on Search). Discovery usually entails problem solving, and

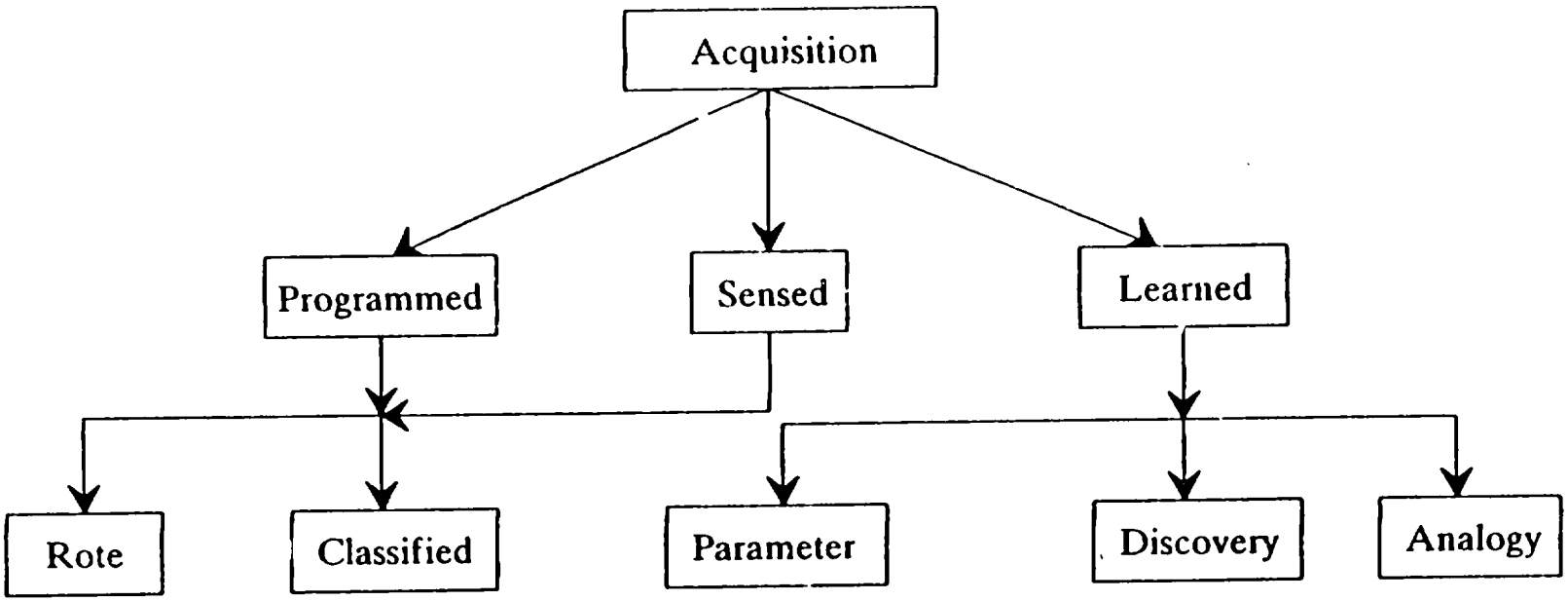


Figure 2. Commonly Used Methods of Acquiring Information

therefore relies heavily on reasoning. Reasoning also underlies learning-by-analogy which involves filling in missing information about some entity if it is known that the partially defined entity is "like" some already known entity. The representation of knowledge via frames and scripts (discussed in Section II.C) facilitates analogical reasoning since one can readily pass attributes between two frames or scripts that are said to be alike.

Retrieval is a very real problem for AI systems for several reasons, not the least of which is due to the large sizes of the data bases. Not only do these data bases contain the collection of facts that are relevant to the problem domain of a particular system, but they contain rules that enable the intelligent manipulation of the facts. One common technique of retrieval utilizes multiple indices which tag facts by one or more of their attributes. For example, a holographic lens could be indexed by such attributes as "optical", "diffraction grating", "conformal", "narrowband", and "lightweight". The LISP programming language, which is the most widely used language in the AI community, facilitates the assignment of attributes to objects in that it centers around lists of related symbols. The above example could be encoded in LISP using a "property list" as:

```
(holographiclens optical  diffractiongrating  conformal
 narrowband  lightweight).
```

Two other retrieval schemes are based on pattern matching and contexts. The pattern matching scheme retrieves data according to some pattern which is related to data categories. A more advanced scheme is contextual storage, in which data are retrieved according to meanings. As an example of pattern matching, consider a data base containing the following lists:

```
(lightsource  laser  heliumneon  wavelength(x) . . .)
(lightsource  laser  NdYag  . . . )
(lightsource  laser  diode . . . )
(lightsource  arclamp  mercury . . . )
```

```
(lightsource  arclamp  xenon . . . )
(powersource  110Vinput  12Voutput . . . )
```

In order to retrieve those elements representing light sources, one could query the data base with a pattern denoted as:

```
(lightsource  ?x)
```

To find laser entries, the pattern matcher would be specified by:

```
(lightsource  laser  ?x)
```

All of these retrieval schemes differ significantly from those used in numeric computers which store data according to memory addresses. Numeric information, being a subset of symbolic information, can be stored and retrieved via the schemes mentioned above, although likely not as efficiently. For example, the simple list (+ 2 4 9) associates the numeric symbols 2, 4, and 9 with the summation operation, and the nested lists (+ (* 3 4) (* 6 3)) associates 3 and 4 with one product operation, 6 and 3 with the other product operation, and associates 12 and 18 with the summation operation. This example has used basic LISP notation in which the first element of the list represents the operation to be performed while the remaining elements are the arguments to be operated upon.

In any discussion of retrieval, one must go far beyond the fundamental techniques for recognizing relevant data in the knowledge base to a discussion of how one searches for the set or sets of data that can lead to a defined goal. However, a discussion of search will be delayed until after knowledge representation is discussed since the two are closely related; i.e., knowledge is usually represented in such a way as to facilitate the search process.

Reasoning, which is the third capability of knowledge systems, is required when the system needs information that cannot be retrieved directly from the knowledge base. AI systems can tradeoff between large

knowledge bases and complex reasoning procedures. Systems must either have a high degree of reasoning power or be able to store and retrieve all relevant information; however, a system that spends too much time searching for reasoning strategies does not have enough knowledge.

Reasoning may be viewed in terms of a movement in a state space in which the states represent all possible situations and the movement is from an initial state(s), representing the current situation(s), to a goal state(s). The reasoning process in solving practical problems typically involves passing through many intermediate states. The allowable transitions between the states are specified either by rules, such as "if..., then..." statements, or via a linking of facts, such as a directed graph. The discussion in the next section on knowledge representation will present various techniques used for state specifications and interstate transitions. A classic example of the state space concept is the game of chess, in which the initial state would be the starting locations of all pieces, and the goal states would be any configuration of the pieces in which all possible moves by the opponent are illegal and the opponent's king is in check. The state transitions would be the rules governing the allowable moves of each piece in each state (each state would have different allowable moves depending on the locations of the other pieces and the proximity of pieces to the edge of the board).

Residence in a given state will most likely present the problem solver with a multitude of possible transitions to other states enroute to a solution. Therefore, search operations also play a major role in the reasoning process. Here, the search is for the path or paths through the state space of a given problem domain, whereas in retrieval it was a search for relevant data in the knowledge base. As stated above, search operations will be discussed following the next section on knowledge representation.

B. KNOWLEDGE REPRESENTATION

Fundamental to each of the types of knowledge discussed above is the problem of how to represent them in a computer in such a way as to

facilitate their interaction, thus producing useful systems. Numerous representations have evolved but most are variations or combinations of the following four: semantic nets, production systems, frames, and logic systems. Semantic networks are very diverse in nature but are generally characterized as being graphical representation schemes in which the graph nodes represent objects or concepts and the links represent inference procedures that relate the nodes. Figure 3 illustrates a very simplistic semantic network that could facilitate arriving at the inference that a Bragg cell has both transverse and longitudinal wave propagation. This type of knowledge is often referred to as declarative, since it is usually derived from factual statements of specific knowledge or relationships.

In their most elementary form, production systems represent knowledge by rules (productions) formulated as "pattern/action" pairs expressed as "If/then" statements. If the "pattern" segment of the statement is true (also known as the antecedent), then the action segment is "fired"; i.e., the state of the machine is modified according to the action specified. Examples of such statements would be: "If the light source is a laser, then it emits coherent illumination," and "If low cost is important and if coherency is not required, then use LEDs instead of laser diodes." Production systems are popular as knowledge representation schemes for expert systems (to be covered in Section II.E), and are in many cases referred to as procedural knowledge, since some action typically results from a rule firing. It should be noted that pattern matching (correlation) plays an important role in production systems in that rule firings are based on "matches" between the rule antecedents (the "if" parts) and the problem states; that is, the matching process determines whether the antecedent is true or false.

Representation via frames involves organizing data by functional groups of hierarchically linked attribute-value pairs. Such a representation is advantageous when dealing with stereotypical concepts such as illustrated in Figure 4. This frame, representing knowledge of a two-dimensional light modulator, might have been referenced by one of several other frames such as ones for optical processing, input/output devices, or

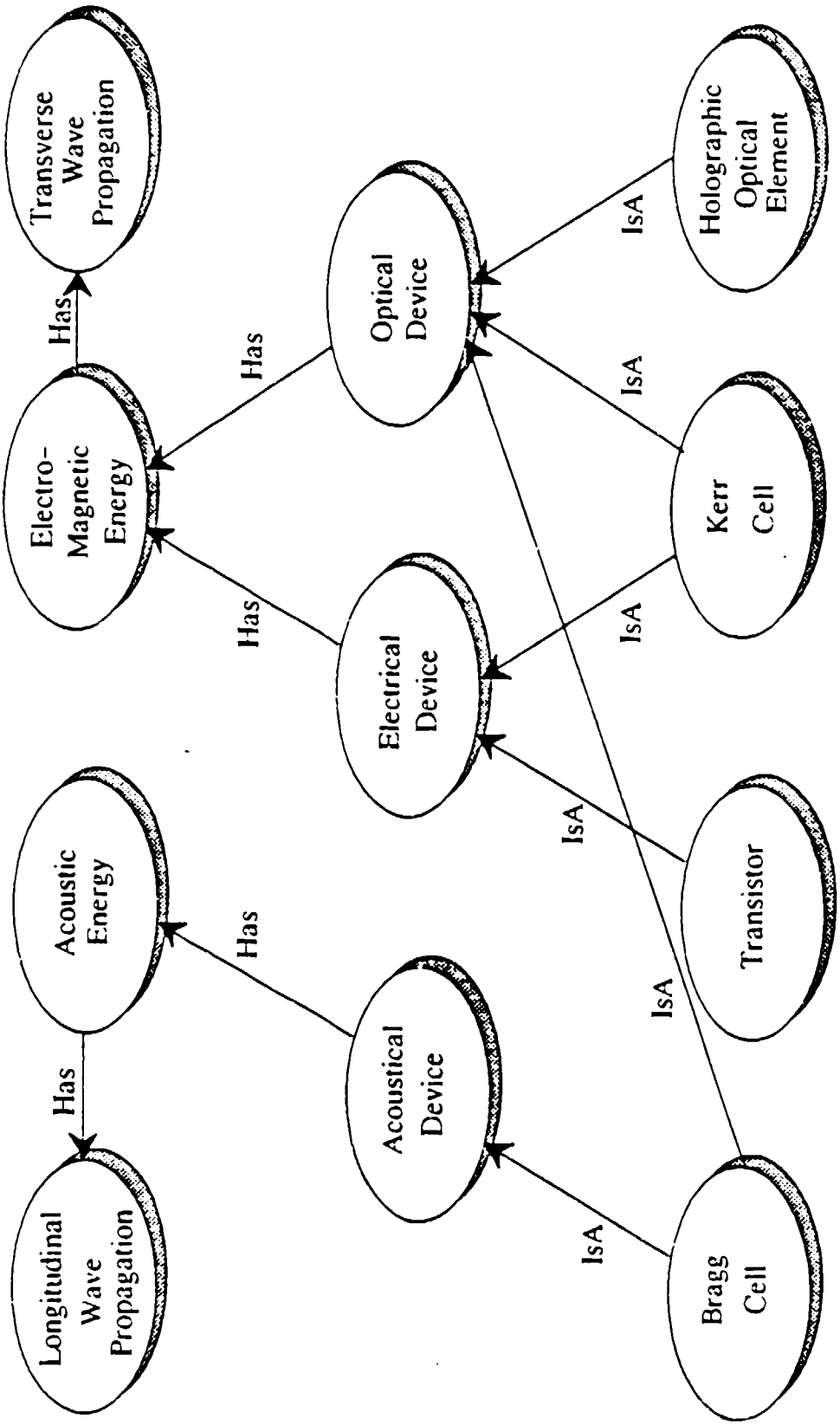


Figure 3. Simplistic Example of Semantic Network

2D Spatial Light Modulator

Specific attributed of 2D SLM's that do not lead to additional frames

Photoconductor:	Silicon
Read/Write Mechanisms:	Optical
Nonlinear Optical Materials:	BSO
Available Sources:	Diode Laser
Frame Rate:	1 kHz
Number of Pixels:	$10^3 \times 10^3$

Figure 4. Frame of Knowledge Representing a 2D Spatial Light Modulator

optical devices. In turn, it references other dependent frames corresponding to each entry in the light modulator frame that has external linkage (shown in the figure by arrows). For example, the non-linear optical material attribute would lead to a frame of information on attributes of such materials. Related to the concept of frames is that of scripts which involve a collection of common sequences of events just as frames involve a collection of related objects and attributes. For example, one could formulate a script for fabricating a light modulator from its component parts.

Logic representation schemes attempt to construct knowledge by the syntactic manipulation of formulas to arrive at TRUE or FALSE premises. The propositional calculus of numerical computing has been extended to provide the formalism known as predicate calculus. Logic is an appealing representation for systems that frequently encounter knowledge base expansion; e.g., such as in theorem proving. The ability to expand the knowledge base results from the power of mathematical deduction to derive new facts from old ones.

Real world relationships are expressed as predicates and their arguments. Predicates represent relationships between things, and predicate symbols are used to stand for these relationships. The predicate applied to its argument(s) returns either a true or false response. The IS-A link between the Kerr Cell node and the optical device node in the semantic net representation shown in Figure 3 would be represented by an ISOPTICALDEVICE predicate in logic representation. The predicate expression ISOPTICALDEVICE (KERRCELL) would return TRUE while ISOPTICALDEVICE (MAGNET) would return FALSE. Other examples of predicates which could be defined (written with variables as arguments for the purpose of generality) are:

WOMAN (x)	TRUE if x is a woman
EQUALS (x,y)	TRUE if x=y
PRESIDENT (x,y)	TRUE if x is president of y

In predicate calculus, it is possible to combine predicates together with their arguments (rather than propositions such as "it is illuminated") using the connectives of propositional calculus including AND ($\wedge$), OR ($\vee$), NOT ($\sim$), EQUIVALENCE (=), and IMPLIES (-> or). In addition, one must introduce quantifiers that assign ranges to variables. For example, the statement EQUALS (x,y) could mean any one of the following four relationships: all x equals all y, a given x equals a given y, all x equals some value of y, or some value of x is equal to any value of y. The universal quantifier V means that all values of the variable x are to be considered while the existential quantifier x means that some particular value of x is to be considered. For example, if the truth of EQUALS (x,y) is to be based on all values of x being equal to all values of y, then one would express it as V x V y EQUALS (x,y) whereas if the meaning were that the statement is to be TRUE if a specific value of x equals a specific value of y, then one would write x y EQUALS (x,y). Examples of expressions written in predicate calculus notation are:

Light is coherent if it comes from a laser.

Vx (LASERLIGHT (x) - COHERENTLIGHT (x))

Some laser illumination is visible.

x (LASERLIGHT (x) $\wedge$ VISIBLELIGHT (x))

Before introducing the final fundamental element of logic, that of functions, let us review the previously introduced elements. These are summarized in Figure 5. Note that the accepted convention is to express variables in lower case alphanumerics and to express the constants and predicate symbols in the upper case. Functions will also be expressed in the lower case.

Predicates are somewhat limiting since their evaluations return only TRUE or FALSE values. For example, the predicates WOMAN (WIFE), WOMAN

DESCRIPTION	EXAMPLES	COMMENTS
variables	x,y	used for general arguments
constants	KERRCELL, MAGNET	specific assignments to arguments
predicate symbols	ISOPTICALDEVICE ( )	defined so as to represent relationships
connectives	$\wedge$, $\vee$,	link symbolic expressions
quantifiers	$\forall$, $\exists$,	quantify range of variables

Figure 5. Elements of Predicate Calculus

(FLORENCENIGHTENGALE), and PRESIDENT (UNITEDSTATES, GEORGEWASHINGTON) would return TRUE provided that in the latter example PRESIDENT (x,y) was assigned the meaning "y is president of x". WOMAN (GEORGEWASHINGTON) would, of course, return FALSE. Functions, on the other hand, can return objects; therefore, they are used as arguments of predicates. An example would be "wavelength (x)" being defined to retrieve the numeric value of the wavelength associated with x; i.e., "wavelength (REDLIGHT)" would return 0.6 microns and "wavelength (CO2LASER)" would return 10.6 microns. An example of a function as part of a predicate would be VISIBLE (wavelength (CO2LASER)) which would return FALSE since the CO2 laser lases in the near infrared region of the spectrum instead of the visible region.

C. SEARCH

The extensive sizes of the knowledge bases and the state spaces required for solving practical symbolic problems dictate the use of some kind of control strategy for searching either for relevant facts in the knowledge base or for solution paths through the problem state spaces. Figure 6 illustrates the various categories of search, from purely random searching to very domain specific heuristic searching (heuristic implying the use of problem domain knowledge to guide the search). Although this simplified block diagram fits search strategies into specific categories, in actuality one encounters almost a continuum of strategies. The more random that a search is, the longer the time needed to reach the goal. On the other hand, the more heuristic, the more complex are the control procedures. In the limit, the most complex controls would incorporate so much knowledge about the search space that the need to search would be eliminated; i.e., the controls would guide the solution directly toward the goal. Since both extremes are unrealistic, one attempts to find a compromise strategy that is best for the problem at hand. In fact, many AI systems use a combination of general purpose and specific purpose schemes that adapt the overall search toward a solution can adapt to a varying structure of the state space as the search proceeds.

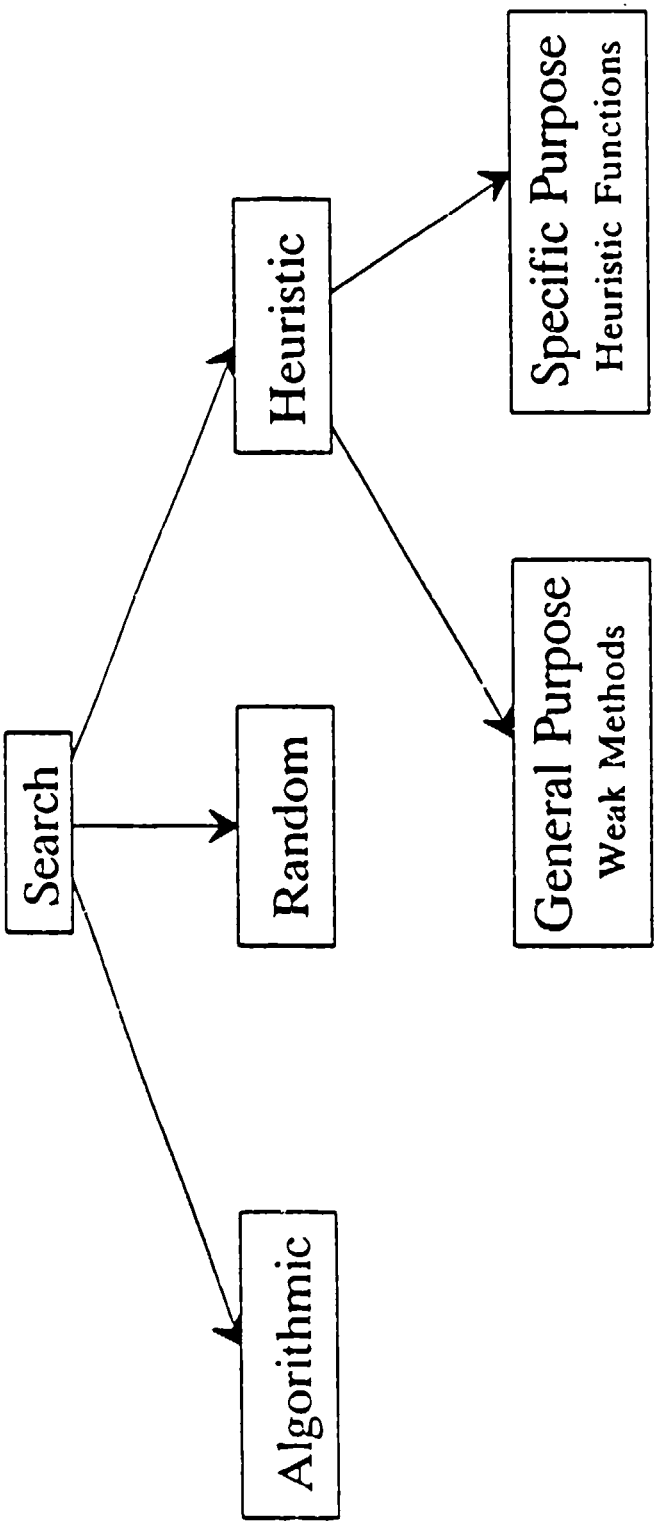


Figure 6. Hierarchy of Search Strategies

Underlying the techniques of search are several general strategies that can be rendered more or less heuristic depending on the degree to which problem domain and goal information are used in selecting various search paths to be tried. Some of these strategies which will be discussed are: tree versus directed-graph, depth-first versus breadth-first, and forward versus backward chaining.

The search process may be accomplished by following either a directed-graph structure, such as illustrated in Figure 7a, or the corresponding tree structure shown in Figure 7b. The directed-graph search remembers all tried strategies so that the search can return to an earlier problem state (graph node) and continue the search along another path. The tree search, on the other hand, may duplicate efforts in pursuing strategies that were already tried. For example, in the push toward a viable solution, the "C" node might have to be expanded a second time (let us say via A C F ...) if the first attempt (via A B E C) did not reach a goal. The obvious disadvantage of the tree is the possibility for redundant effort, but the advantages are the use of much less memory in not having to store the previous strategies and the freedom from running a test on every generated node to determine if it matches a previously generated node. The choice of tree versus graph is often decided by the nature of the problem being solved; i.e., how frequently are repeated states likely to occur. The availability of memory will also be a deciding factor.

Search schemes may also differ with respect to the order in which the nodes are searched. A strictly serial search, known as depth-first, pursues a given strategy (e.g., branch of a tree) until the strategy has either succeeded in reaching a goal or has been shown to terminate in an unsuccessful search. In the latter case, one backtracks to the most recently traversed node that possesses a yet untried branch. This search procedure saves on memory since unsuccessful branches can be discarded; however, if the problem domain is characterized by long search paths, the depth-first search could be exceedingly costly in terms of time. An example of a depth-first search through the tree illustrated in Figure 7b might be A B D B E C A C F G ...

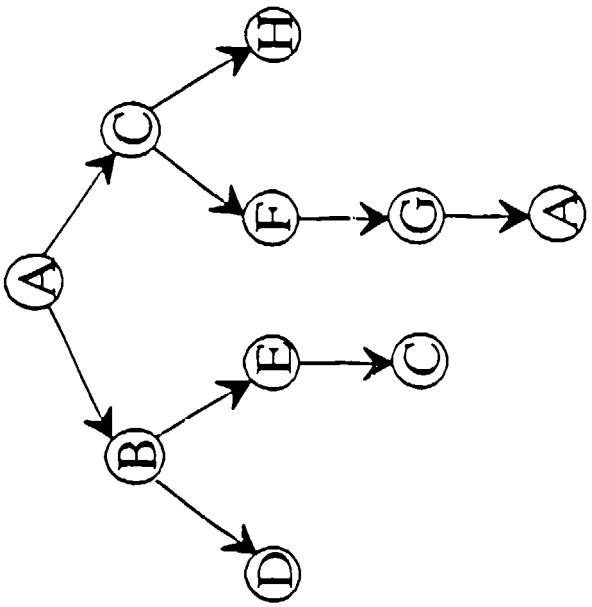


Figure 7b. Tree Search

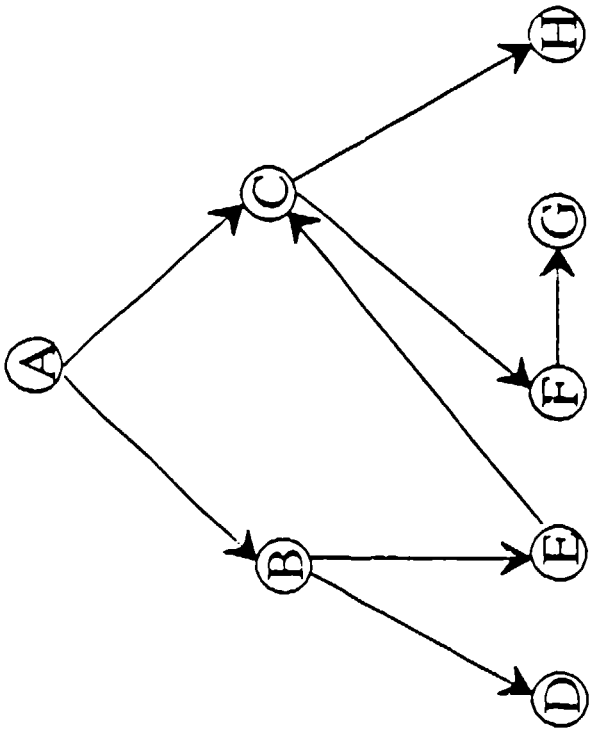


Figure 7a. Directed Graph Search

In the breadth-first search, on the other hand, all states linked to the starting state are tested to see if they match the goal state before proceeding any deeper into the tree or graph. If a goal has not been reached, then all of these first level states (or nodes) are expanded; i.e., all of the linked states are generated and tested. For example, the breadth-first search through the tree of Figure 6b would proceed via the nodal order A B C D E F H C ... until a goal state is reached. This type of search could have great utility in parallel processing systems, such as the types that will be discussed in Section IV.

Up to this point, problem solution and reasoning have been presented as a procession from a starting state or states toward the goal state(s). Another option that we as humans sometimes employ is to start with the goal and proceed backwards in an attempt to satisfy the initial conditions. In backward reasoning, or backward chaining as it is known in the AI community, one first generates one or more states that could produce the goal state and tests to see if a match exists with the initial state(s). If not, the search continues on backwards. In production systems, this means that the "then" parts of the rules are matched and the "if" parts are fired (i.e., each "if" part is used to generate a more forward node). There are two factors which would strongly influence the choice of backward chaining over forward chaining - the branching factor going backward versus going forward, and the number of goal states versus the number of initial states. That is, if the generated search tree branches out significantly more in the forward search than for the backward search for a given problem domain, then backward chaining would be preferable for such problems. If the branching is approximately the same in both directions, the number of goal states versus the number of initial states becomes a deciding factor. Backward chaining looks more appealing in solving synthesis-type problems for which there exists a broad spectrum of objects from which to synthesize. An example would be the determination of which material characteristics are needed to optimally realize a specific device. Here it would be better to start with the goal state (the device requirements) than to start with the sets of characteristics for all possible materials.

Chess, on the other hand, could never be reasoned through via backward chaining due to the extremely large number of ways to attain checkmate.

For many practical problems, the state spaces are so large that searches for guaranteed optimum solutions are sacrificed in favor of incorporating strategy and tactics into constraint of the search process and being satisfied with a good solution rather than necessarily the best. Search processes involved with the playing of chess are an excellent example of this; otherwise, the chess problem could not realistically be solved. As previously mentioned, varying amounts of information can be provided in order to judiciously guide the various search schemes just discussed. Techniques such as indexing, factorization, and template matching fall under the category of general purpose heuristics; however, their effect in constraining complex searches is characteristically weak, often necessitating the use of more complex heuristics in conjunction with these more general purpose ones. Indexing involves using a predetermined scheme to assign indices to problem states and storing the rules applicable to each problem state in such a way that they can be associated with the index for that state. Residence in a particular state can then invoke the index for that state, which, in turn, can call up all of the rules that could apply. More appealing is the operation of factorization. If the knowledge base is divisible into broad sets which have small cross-correlations, entire sections can be ignored by considering appropriate problem domain information. For example, if the question at hand deals with diagnostics for lung diseases, the system does not need to search any part of its knowledge base dealing with procedures for use in actual lung operations. Search constraint can also be achieved via template matching, analogous to classical pattern matching where the matching is used to verify that the correct shape, word, etc. has been found. Template matching is frequently used in speech recognition, natural language understanding, and image understanding, all of which will be discussed in Section III.

One can get more quantitative in search constraint by utilizing some kind of heuristic (or evaluation) function. This falls under the class of specific purpose heuristics as denoted in Figure 6. The heuristic function

generates a weighting or cost factor to be associated with each node that is some measure of the "goodness" of the solution path to that point. Different types of problem domains have different opportunities for defining such functions (thus the notation "specific purpose"); however, frequently used measurement concepts are: a metric representing the length or difficulty of the search to the node in question, or a metric representing the distance to the goal node or difference between the current node and the goal node. As mentioned earlier, there will always be a tradeoff between the search time saved by the heuristic functions versus the time needed to compute the functions themselves; i.e., complex functions may provide excellent guidance for the search, but the time needed to compute them may be more than the time that would have been used by a more random search.

Once a measure function has been defined, the search process can proceed along what is known as a best-first search. Using this technique, the nodes to be expanded at any given point in the search are the ones with the best "goodness" measure. For example, consider the number beside each node in the tree of Figure 8 to be the value of the heuristic function such that the lower the number, the better to expand that node. Then the best-first search would proceed as follows: A C G H B D K E F I J L.

D. ATTRIBUTES OF SYMBOLIC COMPUTING AS COMPARED TO NUMERIC COMPUTING

At this point, the reader should be gaining a cursory view of what constitutes symbolic computation. This understanding can be enhanced further by making direct comparisons between symbolic computing and the more familiar numeric computing. Figure 9 lists many of the attributes of the two computational methods in such a way that a comparison can be made between corresponding attributes in each column. The comparisons are discussed below.

The logical inference is the analogue of the floating point operation. Whereas the floating point operation is basic to the manipulation of numeric symbols, the inference operation is used to manipulate the much

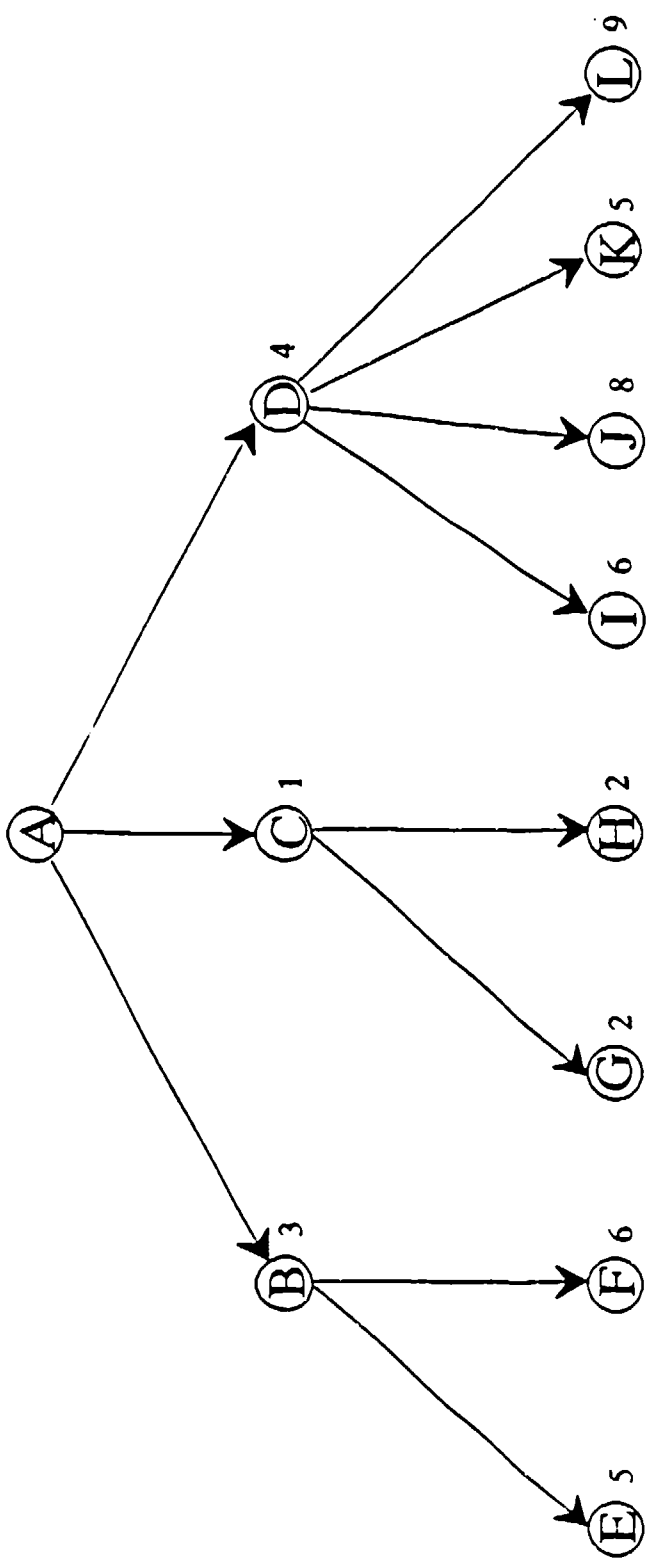


Figure 8. Best-First Search

SYMBOLIC	NUMERIC
Logical Inferences	Floating Point Operations
Dynamic Data Structures	Static Data Structures
Heuristic Search	Algorithmic
Inexact, Incomplete Knowledge Acceptable	Correct Complete Data Required
LISP, PROLOG, ...	FORTRAN, PASCAL, COBOL,...
Independence of Control and Problem Knowledge	Integration of Control and Information
High Level Modification	Macro Level Modification
Explanatory Capabilities	Limited Explanation

Figure 9. Attributes of Symbolic and Numeric Computation

broader class of symbols encountered in symbolic computations. A logical inference is generated by combining knowledge elements or groups of objects to reach a conclusion. Whether the logical inference is done using syllogisms (cascaded "if/then" statements), graphical linking, frame matching, or logical inference techniques depends considerably on the knowledge representation used (production system, semantic network, frames, or first-order logic, respectively). But it is the combination and manipulation of symbolic data structures (e.g., objects and their attributes) by logical inferencing that is the basis for most reasoning techniques in symbolic computation.

The well-structured data formats of vectors, matrices, etc. used in numeric computing give way to data structures that can change their shapes in symbolic processing. In performing tasks such as navigating over unknown terrain, playing a game, or carrying on a dialogue, one knows prior to performance of the task only the general form the data must take to represent the route planning, the responses to the opponents' actions, or the meaning of the spoken phrases, respectively. Therefore, symbolic processors use lists of objects connected by pointers, such as those discussed in previous sections of this chapter. The input data, the data manipulated, and the end point in the manipulation of that data are usually a phrase, a concept, or some other symbolic structure of unconstrained size and/or shape, rather than data structures with specified dimensions as in numeric processing. Further, dynamic data structures allow the machine to deal with inexact information or to arrive at uncertain or inexact conclusions.

This use of inexact or incomplete information gets to the heart of the differences in using symbolic versus numeric data. The algorithms used on numeric computers have little ability to generate correct numerical output from qualitative descriptors or from incomplete numerical input data. They depend upon exact data since all fundamental operations are combinations of numeric entities. On the other hand, it is a feature of most symbolic representations that conclusions can be reached in a qualitative sense by generating relations among the functional attributes. This is possible

even when the input information is incomplete, or, in some cases, inexact. It should be noted that such techniques are at the forefront of today's research in symbolic computation, and comprise a discipline called reasoning with uncertainty (ref. 6).

It should be obvious that in order to take advantage of the power of symbolic computing, such as the use of inexact or incomplete knowledge in reasoning, specialized programming languages are required. We should note, however, that the numeric programming languages, such as FORTRAN, could, in the hands of clever programmers, be used in symbolic computing. However, they have not been optimized for such use and would make symbolic programming a very cumbersome task. The LISP language, and its major derivatives of INTERLISP and MACLISP, create a programming environment that facilitates the manipulation of symbolic expressions characterized by flexible data structures. The semantic meanings of objects are readily changed by adding and deleting the variable lists of attributes. Another notable characteristic of LISP is its recursive nature - any LISP function can call itself and any program can be defined in terms of itself. Such a capability facilitates the search of lists of indefinite length in the search for certain elements (attributes) in the lists.

A very different language is PROLOG (PROgramming LOGic). Based on production rules, it uses pattern matching techniques to prove or disprove program statements. The language relies heavily on predicate calculus in establishing relationships between objects. The fundamental operation in PROLOG is the logical proof of some condition or relationship starting with a set of more primitive conditions.

The sixth element in Figure 9, the independence of control and problem knowledge, represents a drastic difference in the way symbolic systems process information. In symbolic computation, the control refers to any process, explicit or implicit, which governs the order of problem solving activities.[3,4] A key aspect of this occurs in expert systems (discussed in Section III.E) where the actual structuring of the solution strategy can be changed recursively based upon changes in the program's evolution. This is very different from a numeric computation, where changes, even conditional

branches within a program, are input "a priori" to the system. It is this independence of the knowledge base from the control activities that allows an expert system "shell" to be robust enough to be applied to more than one problem domain. As an example, although the operating program MYCIN was originally developed to aid in the medical diagnosis of bacterial infections, it may be applied to a knowledge base of crystallographic information to aid crystal growers. Instead of containing rules relating symptoms, bacteria, and remedies, the knowledge base would contain rules relating measurements, crystallographic structure, and recommendations for regrowth. In the numeric domain, one cannot take a program written for, say, VLSI design and adapt it for lens design with only a change in the input data.

Even the way one makes changes to symbolic programs differs from techniques in the numeric domain. If a change is required in the program of a numeric computer, the entire program, or at least the macro in which the change is to be made, must be pulled up, edited, and returned to the system, at which time the macro or the entire code must be recompiled. In programming environments built upon LISP one can modify the program without such activity. LISP programs and data sets are both written in the same syntax and form. As a result, a LISP program can manipulate (alter) another LISP program or data base. It can automatically modify faulty rules or knowledge, or it can call suspect rules or knowledge to the attention of the operator who can then make changes interactively without having to recompile the entire program. These environments are very powerful, allowing the operator to slide a window through the program which allows one to see in real time the outcome of a change in the knowledge base or rule structure anywhere in that window. At the present time, this form of software engineering is not available in numerical computers, for which the time-consuming debugging cycle is: find bugs, rewrite, recompile, rerun, look for new bugs, rewrite, recompile, rerun, etc.

If making changes to symbolic programs differs from numeric practice, then it is not altogether unexpected that the power of symbolic computing can be used in the debugging phase of program development as well. It is

certainly possible to build into a numeric computer program the ability to print out messages indicating which "if/then" paths were taken during execution of the program. In a symbolic computer, however, not only can the machine trace for the operator the path taken to the solution, but also it can tell the operator why it took each path and not others. This feature is useful for several reasons, only two of which are stated here. First of all, it is a valuable diagnostic tool; in fact, it is an integral part of most development environments designed for use with LISP machines Second, human beings want to know "why?"; that is, we tend to ask how some conclusion was reached by another human in order to form some opinion about its validity; and one would not expect a machine replacing a human expert to be exempt from having to justify its own conclusions. This leads the user to a feeling of confidence in the machine.

Having looked at the similarities and differences between symbolic and numeric computing, the reader can begin to understand some of the ways in which symbolic computing can be applied to real-world problems. The earlier discussions on knowledge representation and search strategies will be built upon in the next section, where the reader will see both how these techniques are specialized and modified as well as how they can lead to performance problems. This is particularly true with the search process, which is very often are the cause for the computational bottlenecks in AI systems, and the discovery of ways in which optics can impact these operations may be a key to optical symbolic computing. Before discussing these opportunities, let us take a look at the symbolic computing domains of speech recognition, vision/image understanding, natural language processing, and expert systems.

CHAPTER III
SYMBOLIC COMPUTATION: FUNCTIONAL CAPABILITIES

The previous section highlighted the underlying principles of symbolic computation, providing both an introduction to the subject and a discussion of the types of representations employed and search strategies utilized. This section will extend that discussion by first presenting an overview of the four main disciplines which comprise symbolic computation: speech understanding, vision, natural language understanding, and expert systems. These applications can take several forms, each of which draws upon aspects of knowledge acquisition, reasoning, and retrieval. If they have one aspect in common, though, it is the use of knowledge, about the system or domain under consideration, to improve the machine's understanding of input information.

A. OVERVIEW OF SYMBOLIC COMPUTING DOMAINS

As in any new field, there is considerable debate as to what constitutes AI and what the appropriate taxonomy of AI disciplines should be. Complicating matters is the fact that the overall nature of the programming tasks and the optimal computing structures for it are still not well understood.[1-3] Nevertheless, major advances have been made in applying the knowledge-based techniques presented in the previous section to a wide variety of problems. As a result, symbolic computing research is currently concentrated in four areas which form the basis of generic machine understanding capabilities: speech recognition and understanding, vision or image understanding, natural language understanding, and expert systems and reasoning.

To enable a machine to respond and identify spoken language, we can apply advanced pattern recognition techniques to process the input signal and recognize the words. This aspect of speech research is termed speech recognition. Language is typically entered into a speech recognition system by means of a microphone.[4] Recognition can occur in any of several

ways, but each typically involves performing a digital comparison of the input phrase or sentence with elements stored in the computer's memory. Speech understanding, on the other hand, focuses on the use of knowledge to attach meaning to a series of spoken inputs. Although no real boundaries exist, in most cases speech recognition is referred to as low-level speech, emphasizing signal processing and template matching; speech understanding, by virtue of its reliance upon knowledge processing, is termed high-level speech. The goal of all speech research, obviously, is to achieve totally speaker independent, high accuracy recognition, over a large vocabulary base.

Vision, or image understanding, as it is more commonly known in computer science, refers to the ability of a machine or computer to understand scenes utilizing a visual input. Such systems have the goal[5] of pattern and image understanding with a degree of accuracy that parallels human vision systems. Recognition can be accomplished in many ways, but most involve matching elements of an observed scene to objects represented in the system's knowledge base. This is similar to the problem for speech understanding systems, except that the information is comprised of input images rather than waveforms, and the objects themselves are three dimensional. Since most visual input devices are two-dimensional imaging arrays, such as solid-state TV cameras, much emphasis has been placed on achieving some level of full three-dimensional information from the input scene. By analogy with speech, image processing functions such as preprocessing, image restoration, or gradient calculations are termed low-level vision. Any processing requiring interactions with the knowledge base are known as high-level vision.

Natural language understanding focuses on the ability of a machine to attach meaning to English language (or some other written language) phrases input to it through a peripheral device, typically a keyboard (see Figure 10). To be effective, therefore, natural language understanding systems should incorporate knowledge of linguistic theory, such as sentence decomposition according to the syntax of the language (parsing) and assigning meaning to words or phrases (semantics).[6] Thus natural language

Figure 10. A Natural Language Understanding System

processing is a very knowledge intensive activity, relying upon knowledge of the grammar and the context to attach meaning to the input sentence or phrase. Knowledge of these and other linguistic attributes must be included within the system's knowledge base, and the challenge, as in other AI disciplines, lies in resolving and implementing one of many different strategies for interpreting input, such as English sentences.

Expert systems are a discipline within applied artificial intelligence which seek to emulate human expertise in specialized areas, known in AI parlance as domains. Examples include the interpretation of spectral data (the DENDRAL project),[7] the configuration of minicomputers from components (the R1 project),[8] and the diagnosis of diseases in internal medicine (the MYCIN project).[9] These computer systems "achieve high levels of performance in task areas that, for human beings, require years of special education and training."[10] Here, expertise is defined as the "set of capabilities that underlies the high performance of human experts, including extensive domain knowledge, heuristic rules that simplify and improve approaches to problem solving, metaknowledge and metacognition, and compiled forms of behavior that afford great economy in skilled performance."[10] (The prefix meta- refers to knowledge about the root word.) These systems have been very successful recently[10] in solving problems and tasks which are knowledge and heuristic intensive, those that would typically take a human expert between 8 and 40 hours to accomplish. Using the capabilities of LISP machines (see Figure 11) and specialized software development tools, expert systems research seeks to capture human expertise accurately enough to apply it to more complex, demanding, and diverse tasks.

These functional capabilities are presented in this order for several reasons. Historically, the first successful applications of AI research centered on the human/computer interface, either spoken or visual, and on game playing by machines. (For a nice review of early AI research, the reader is referred to reference 6, Vol. 1) This has evolved into what we now know as low-level processing, comprised mainly of signal and image processing, with knowledge based techniques naturally assigned the role of

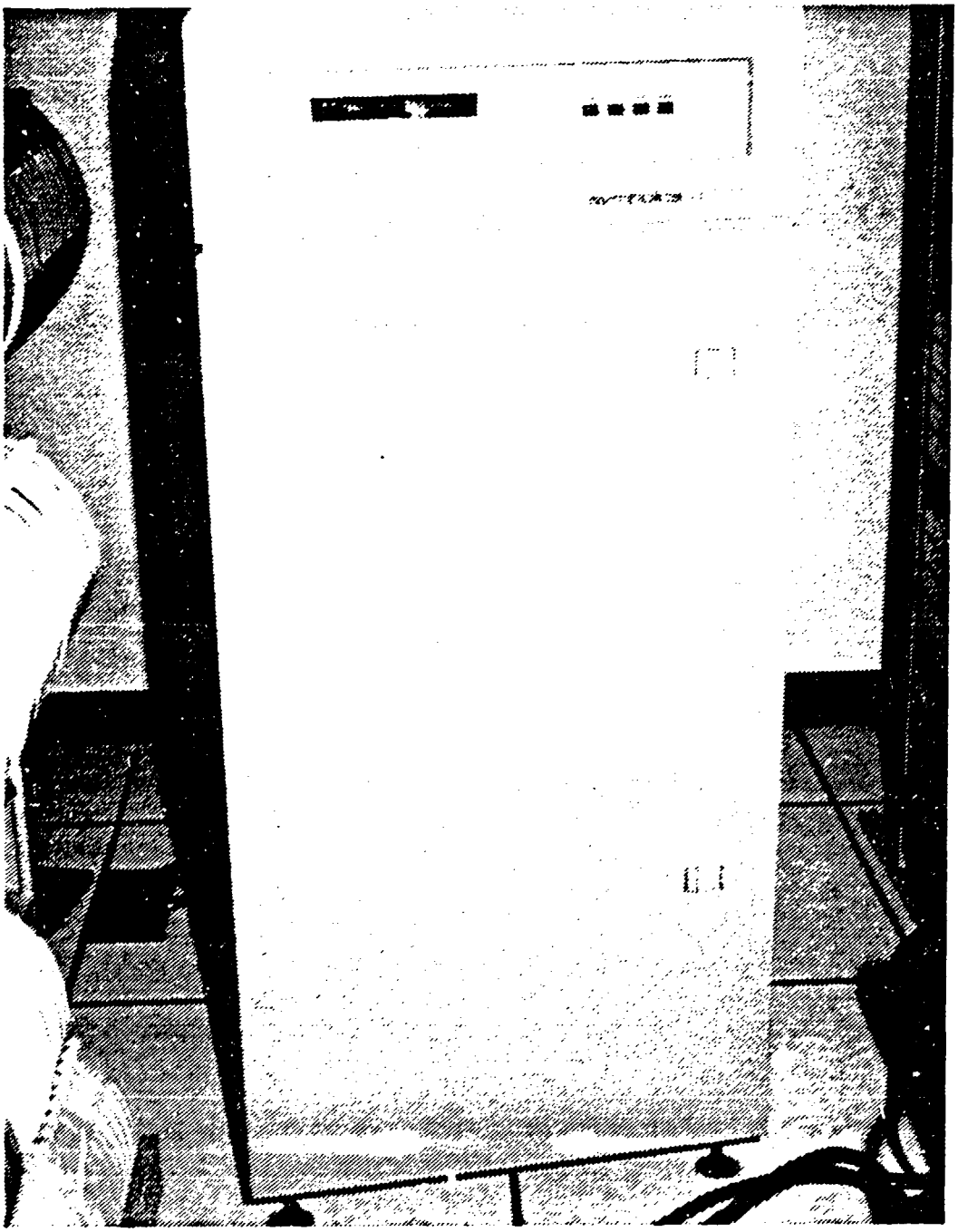


Figure 11. A LISP Machine

(Courtesy of Symbolics, Inc)

higher level processing. By virtue of their long history, speech and vision research are thought to be more mature discipline than natural language or expert systems. As an example, many of the concepts required for natural language processing had their genesis in linguistic theory and in higher level speech research. And the explanation facilities in expert systems have in turn developed out of natural language research.

Second, there is a natural evolution from capabilities which primarily focus on the man-machine interface to those which emphasize computational reasoning with minimal human interaction. Finally, this will allow us to minimize the amount of overlap and repetition of concepts. In the ensuing discussions, we will see a great deal of commonality between each of these disciplines, and in many cases this is a result of critical ideas transitioning from one area to another. The use of the blackboard architecture (to be discussed in Section III.B) is a case in point. Originally developed for the Hearsay speech understanding system,[4] it has now been successfully applied in natural language and expert systems.

In what follows, each of these symbolic computing capabilities will be discussed in greater detail, with an eye towards identifying the underlying technology. In doing this, the major challenges to current research in each field will be highlighted. Each subsection will conclude with a review of current and projected machine intelligence capabilities, and an analysis of the problems and computational bottlenecks impeding the progress of each AI research area.

B. SPEECH UNDERSTANDING

Most speech understanding activities attempt to endow computer systems with the required knowledge and auditory discrimination to achieve real-time machine understanding of continuous spoken input. Unfortunately for computer systems, humans have not standardized on any particular form of word pronunciation. Thus, even systems which seek to recognize speech input (but not necessarily understand it) are faced with the problem of distinguishing words independent of speaker, the rate of speech, any

dialects or accents, vocabulary, dropped syllables, and last, but not least, background noise environment.

This is obviously a formidable problem, since humans themselves do not yet have a reliable, 100 percent speaker independent speech understanding system. We have problems understanding speech in which adjacent words are run together, influencing the pronunciation, as is typical with geographic references such as Long Island, which is often spoken as Longoilan. We still have trouble with different dialects, e.g., the word oil in various sections of the country is pronounced as all or as ole, and very often we have to infer words or missing phrases from context. This last point embodies the critical challenge for knowledge retrieval and reasoning in speech systems - the use of pattern matching techniques for word identification, and knowledge for understanding meaning and for interpretation. The question is how to apply the knowledge, and when (see Figure 12).

Many techniques have been developed for knowledge-based interaction in speech, but the two most developed are the isolated word recognition (IWR) systems and the continuous speech understanding (CSU) systems.[11] The isolated word systems focus in on word identification, using segmented spoken input, such as:

"helium-neon...laser...broken"

or:

"weather...today...rain"

to decrease problems associated with identifying the beginning and ends of words. The problem of identifying the word from the transformed acoustic input still remains, but is handled by template matching. This process, shown in Figure 13, involves taking the input signal and identifying key features, such as the major sound variations and the beginnings and ends of words. The next step is dynamic time warping, which seeks to align the

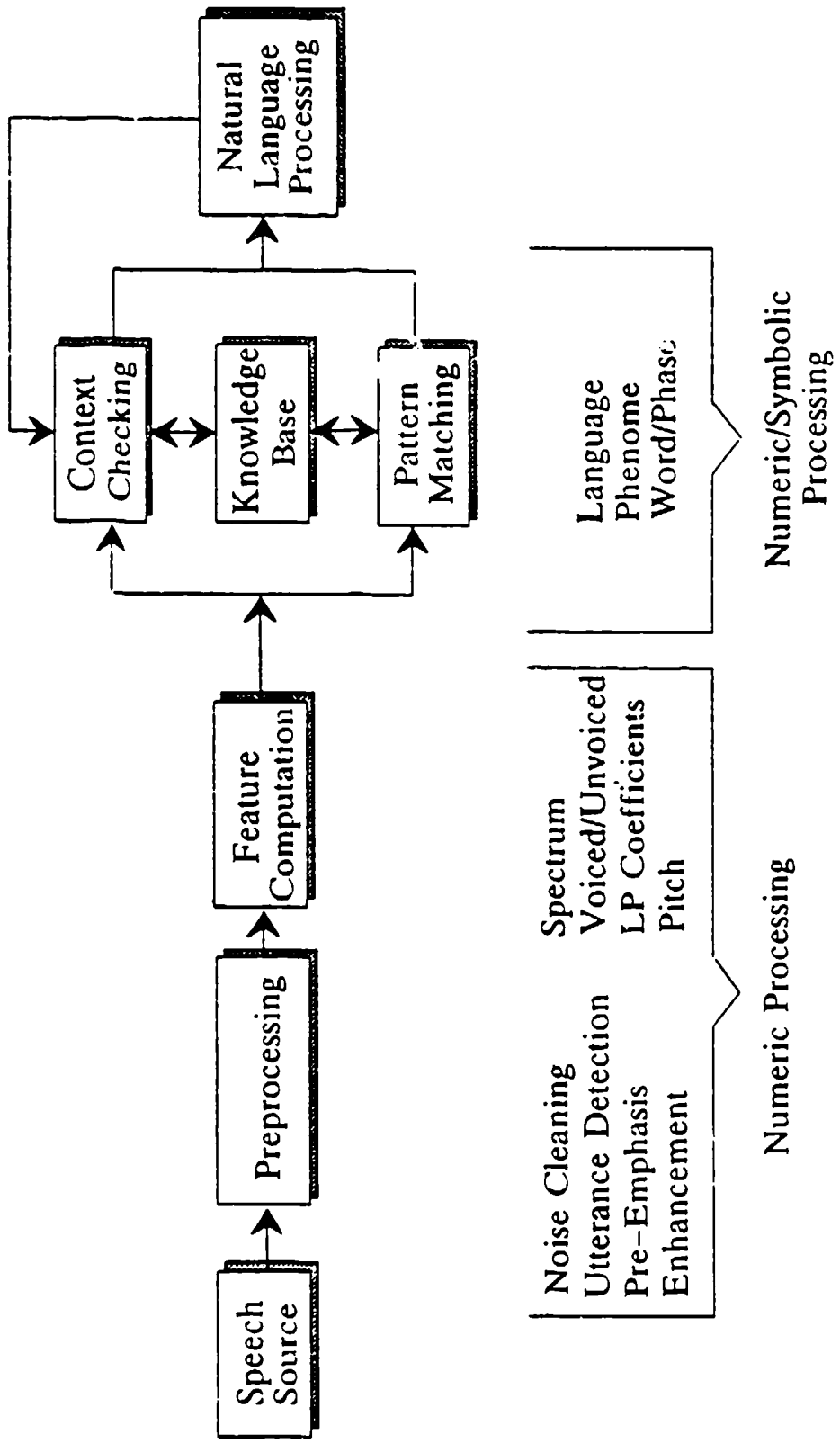
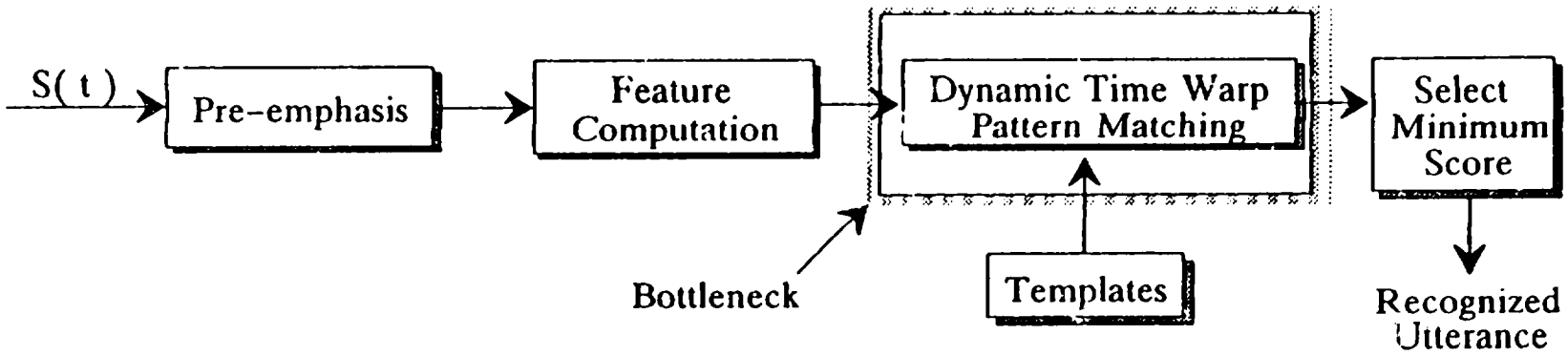


Figure 12. Speech Understanding Paradigm



Pattern Matching Metrics: $\sum_{i=1}^{N_F}(D_i - T_i)^2 W_i$ (Weighted Euclidian for Filter Bank Features)

Number of Features $= N_F \sim 8\text{-}15$

$$\frac{\sum_{j=1}^{N_F}\sum_{i=1}^{N_F} D_i R_{ij} D_j}{\sum_{j=1}^{N_F}\sum_{i=1}^{N_F} T_i R_{ij} T_j}$$ (Itakura Metric)

Figure 13. Low-Level Speech Processing

length of the spoken word with some internal reference length. As an example, the words helium-neon and Caribbean are often pronounced as:

"helium-neon"	"Care-ih-be-yan"
"heel-e-yum nee-un"	"Cah-rib-e-an"
"heelyum-neun"	"Cah-rib-be-yun"

each of which is spoken at a different rate. To standardize on the lengths of words, the signal, as a function of time, is stretched or compressed, in order to make it compatible with an internal reference length. The features of this "standardized word" are then compared with templates stored in memory, and weighted metrics, such as those shown in the figure, are used to match the word with its counterpart in memory.

It is here that we see the greatest bottlenecks in the low-level process. To give an idea of the computation rate, there are approximately 500 metric computations typically performed per vocabulary word using dynamic time warping during a 500 msec utterance.[13] Using this as a basis, the total number of multiplies (or inner product computations) per word ranges from 5 K to 10 K, using the weighted Euclidean metric for filter bank features[14] and the Itakura metric for linear predictive coding (LPC) features.[15] Therefore, an optimal inner product processing environment, such as optics, should have great utility in speech recognition.

The template match is not the only computational bottleneck, unfortunately. The application of high-level knowledge, which is a retrieval and reasoning problem, also limits the accuracy and timeliness of the system, preventing real-time operation. Figure 14 gives an example of how knowledge is used to obtain meaning from the input phrase. Here, successive matchings with words in the network allow the machine to interpret the phrase. The reader should note that this is just one of a number of possible representation techniques, and speech systems have evolved which have used frames, scripts, and rules as well as semantic networks.

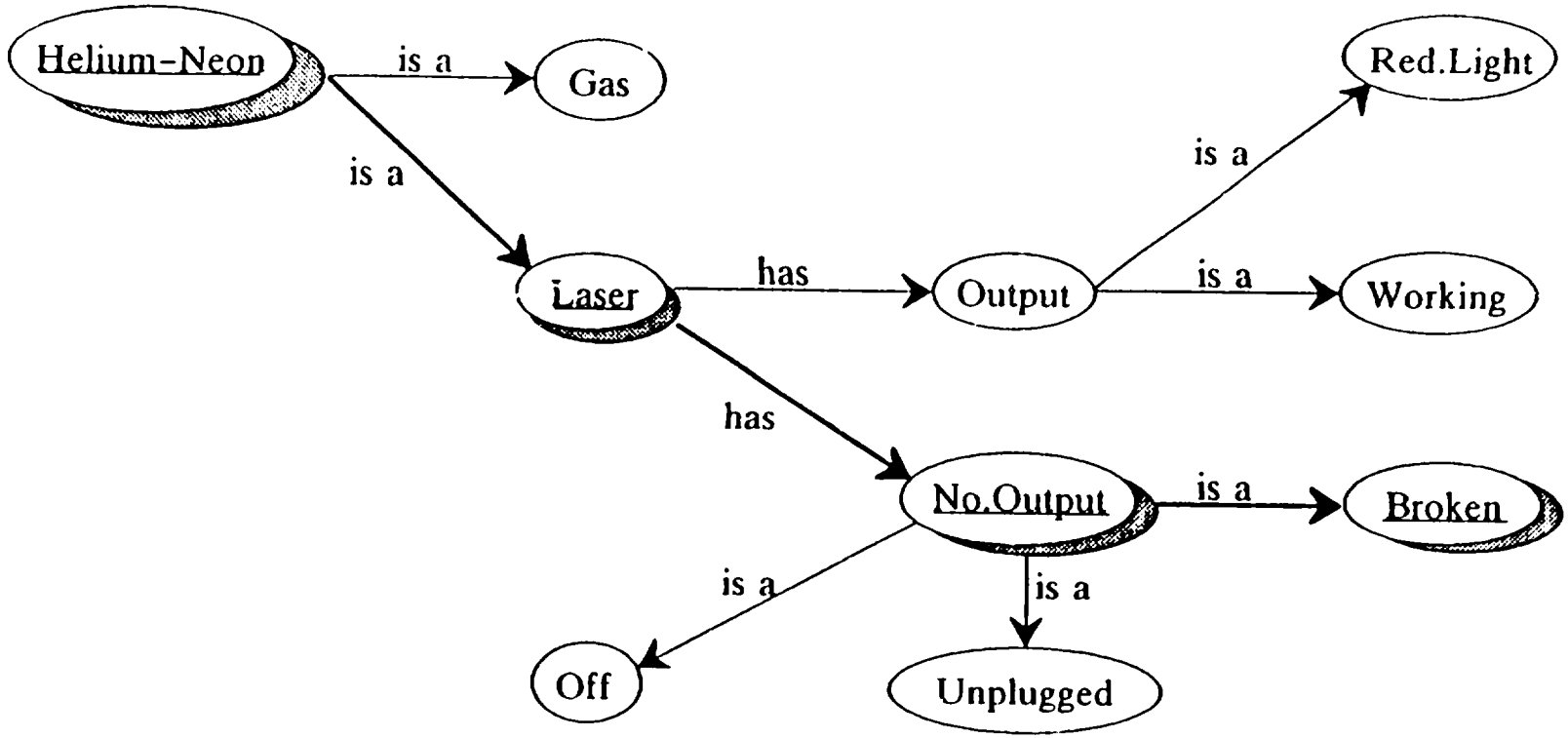


Figure 14. Semantic Net for Speech Recognition

There are many types of knowledge that can be used in understanding speech. Many of these are derived from linguistic considerations, and we can therefore expect that they will be important in the discussion of natural language as well. Each of these is specific to the vocabulary under consideration, and is heavily dependent upon the rules associated with the grammatical structure of the language. Thus it is important for the machine to understand that adjectives modify nouns, adverbs modify verbs, ...etc.

The most obvious, and probably the most familiar type of knowledge, is phonetics, which refers to the physical characteristics of the sounds of each word in the vocabulary; it is the acoustic signature of the words. Another important type of knowledge is morphology, which refers to the ways in which the basic units of words (basic morphemes) can be combined to form new words, plurals, tenses, etc. Using these types of knowledge as a basis, higher level knowledge attributes can be used to determine meaning from spoken input. In the introduction to this section, we introduced the concepts of syntax (the sentence structure and grammar) and semantics (the ways in which word meanings are combined to form the meanings of sentences and phrases) in the discussion on natural language. It should be clear that these types of knowledge are necessary for any speech recognition system as well.

The last type of knowledge, and possibly the most important for high-level speech, is pragmatics. Pragmatics refers to the rules of conversation and dialogue, which allows the system to distinguish questions and queries from factual statements. In the following examples, the statements all have the same syntactic and semantic meaning, yet each represents a different type of interaction with the machine:

(1) "The part in the helium-neon laser is broken."

(2) "Is the part in the helium-neon laser broken?"

(3) "What part of the helium-neon laser is broken?"

The first is a simple factual statement, whereas the second is a rearrangement of the first, requiring a yes or no response from the system. In the third, however, a yes or no response is insufficient; another level of

response is called for from the system. It is the pragmatic knowledge of the vocabulary that allows the system to recognize and understand differences in sentence and phrase meanings.

These types of knowledge could have been incorporated into our earlier IWR example, but they were not required. The scheme for using knowledge in the IWR example is very general and is applicable to most types of knowledge, and is referred to as the bottom-up hierarchical paradigm. Here, bottom-up implies the use of numerical techniques at the start of the algorithm to improve the signal and perform all feature extractions, which collectively are known as low-level speech processing (see Figure 12). The high-level processing is the use of knowledge, and in this case occurs after initial low-level processing has been completed. Hierarchical refers to the passage of system control from the low-level, through a series of intermediate, sequential steps, to the point where the knowledge interactions occur.

This paradigm is not unique, and as we shall see in our discussion of continuous speech (below), can be structured as a top-down strategy, or even a middle-out scheme. In each case, however, the overriding consideration is where and when the knowledge about the problem is retrieved and utilized. If it is at the beginning, chances are that we are dealing with a top-down paradigm; if it is at the end, as in the case of the isolated word example, it is a bottom-up scheme.

In continuous speech understanding, grammatic or linguistic models are used to constrain the knowledge retrieval process, which means limiting the active vocabulary or the number of words possible at any instant in time in a CSU system. Not only is processing time saved by limiting the size of the search space for each word, but the potential for confusion is lowered and a higher recognition rate results.

The basic CSU system consists of an acoustic processor, which carries out the initial signal to symbol transformations, and a linguistic decoder, which applies knowledge to understand the spoken input. Typical CSU systems take two forms: recognizing individual words in random arrangements, such as for data base retrieval, and understanding meaningful,

continuously spoken sentences. The two have some features in common with IWR, as well as a few differences, but the principal difference is in the area of word segmentation.

Word segmentation, or determining the beginning and end points of individual words, is a critical problem in CSU. Since words spoken in continuous speech tend to run together, the end-point detection algorithms used in IWR will not work. Several methods exist for overcoming this problem, and all are variations on the dynamic time warping algorithm discussed earlier. This still remains as a processing bottleneck, although not as severe as the template match or the knowledge retrieval process.

Most continuous speech understanding systems are top-down, knowledge directed systems, using knowledge about the speaker's problem to limit or constrain the expected content of the input.11 Such a technique has been successfully implemented in the Hearsay-III system,[12] where each type of knowledge can interact with the partially processed input signal independently of the others, as shown in Figure 15. The novel feature of this system's architecture is the blackboard, an area in memory allocated for communicating between the various knowledge "experts." In fact, by allowing each type of knowledge to operate on the input signal, we have in fact developed individual speech "expert systems" (see Section III.E). This blackboard concept, which allows cooperation among multiple knowledge bases, has been successfully applied to both natural language understanding systems and to expert systems.

The use of heuristic search to constrain the input is not without its price, and that is the tradeoff between generality and performance. This will be a theme that we will see throughout this section, and one which underlies all AI research at the present time. To appreciate this tradeoff, the reader should recognize that the vocabulary of the English language is very large - the 24 volumes of the Oxford English Dictionary are a testament to that. Yet most human beings have a usable vocabulary must smaller than that, but even 20,000 words are staggering when one looks at the number of possible combinations and pronunciations associated with each word. Early speech systems, which sought to include most words within

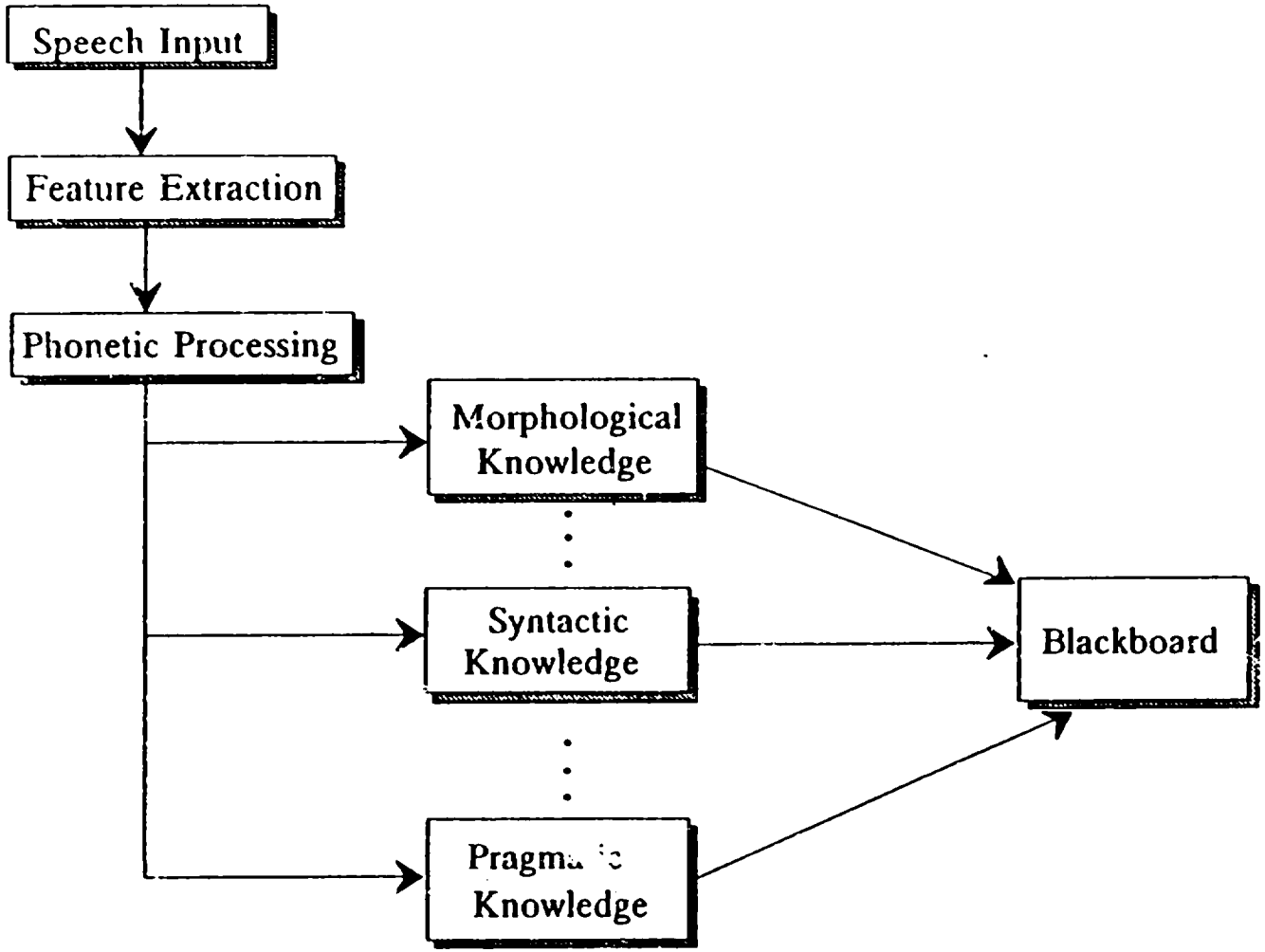


Figure 15. Architecture of the Hearsay III Continuous Speech Understanding System

the typical human vocabulary, were never able to even approximate real time operation, and spent far too much time searching memory. The solution to this bottleneck, borrowing from cognitive psychology, is the use of knowledge about the problem or domain to constrain the search. Once this was realized, near-real-time speech systems became possible, as well as the recent successes of vision, natural language and expert systems.

The reader should note that much of the recent improvement in the performance of both IWR and CSU systems is due to developments in special purpose signal processors. These include special purpose IC's, as well as systolic structures for processing lower level speech. The remaining critical bottleneck, that of knowledge retrieval and processing, requires additional research into symbolic computing, in order to structure an optimal high-level speech processing environment. (Several such structures are suggested in Section IV, where the relationship between processing structure and function will be examined more closely.) Special purpose processors are also under consideration for vision systems, as we shall see in the next section.

C. VISION

Vision, as its name implies, involves providing a machine with the capability to understand visual input in real time. As in human vision systems, machine vision includes the identification of what is in the image or scene, and how the various elements are related to one another, both spatially and temporally. Such systems have broad application, both commercially and militarily, beyond their use as an input device to ease man-machine communications. Many of the recent advances in process automation and robotics (e.g., bin-picking robots) have become possible because of machine vision research. Other commercial sector applications include: sensors for automated welding, handling hazardous materials, VLSI manufacturing, computer aided design and manufacturing (CAD/CAM); inspection of manufactured goods; medical imaging; remote sensing for cartography, traffic monitoring, driving aids, land use management, and

exploration for oil and minerals. Potential military applications are just as diverse, such as autonomous vehicle navigation, photointerpretation, reconnaissance, target acquisition and range finding, terminal homing, and tracking of moving objects. We also see it as the most obvious application for optics, since visual information is inherently optical in nature and can be processed two-dimensionally.

In many ways, the techniques used in machine vision research build on the knowledge-based systems presented in the previous section on speech understanding. As in speech, high-level knowledge about the scene or image under consideration is effectively use to constrain the object identification and understanding process. Similarly, in vision, this high-level knowledge specific to the scene can be effectively used in guiding lower level operations. Later on in this section we will see examples of how knowledge can be applied to constrain the processing.

Another parallel between speech and vision is the approach to use of knowledge and control. Early work in vision was carried out using the bottom-up hierarchical paradigm, similar to the case in isolated word recognition systems. With this approach, preliminary processing performs edge detection, feature extraction, and linking without the benefit of knowledge-based inferencing. As in the case of speech, this preliminary processing is referred to low-level vision, and the subsequent interaction with the knowledge base is termed high-level vision processing.

Currently, developers are exploring advantages of mixing the high- and low-level processing in paradigms of broader scope. For example, high-level reasoning about expected features and their relationships is useful in tasking and guiding the lower level processing. The utility of this stems from the efficiency of focusing on specific regions of an image for a specific purpose, such as resolving an edge and following a lineal feature. At other stages in the vision process, the low-level processing may be involved in generating hypotheses to be evaluated by the top level processes. In this case, the efficiency of the process is improved by avoiding hypotheses which are inconsistent with the image.

Finally, just as speech systems incorporate pattern recognition techniques in the low-level processing, vision research has also developed as an outgrowth of early pattern recognition work. However, treating vision as mere pattern recognition is inaccurate; pattern recognition is strictly numerical in its approach, operating directly on the image. Vision, in contrast, uses knowledge about the scene to categorize the image, and operates on the symbolic representation of the image rather than the image itself. This is a major and important distinction, and is analogous to the use of syntactic knowledge in understanding language (see natural language understanding in Section III.D).

Having introduced the concept of machine vision, we would like to describe, in an elementary fashion, the types of processes and computations associated with vision. Detailed discussions are beyond the scope of this text, but the reader is referred to excellent books on the subject by Marr[5] and by Ballard and Brown[13] for additional information.

For simplicity, in the following discussion of vision, the processing will be considered to originate with the pixels on the visual detector, which is usually some form of two dimensional imaging array, such as a TV camera or vidicon. As in speech, low-level processing focuses in on the transformation from signal-level to symbolic information. This information is in the form of intensity variations across a two dimensional array, and the relative positioning of those variations. Color and texture information can also be utilized, since both can be extracted from the input image. The processing procedure begins with extraction of features via the processing of pixels to find edges and regions, as shown in Figure 16.

The first step in this is the the preprocessing stage, which prepares an image for actual image processing For example, image restoration may be required to remove the effects of noise and optical or motion blur. Geometric distortion correction can be required to remove the effects of angle of view, to correct for common lens distortions, or other types of distortions introduced in the process of presenting the image to the computer. More than one sensor may be used in order to obtain, for instance, a 3D image, or to take advantage of information in several

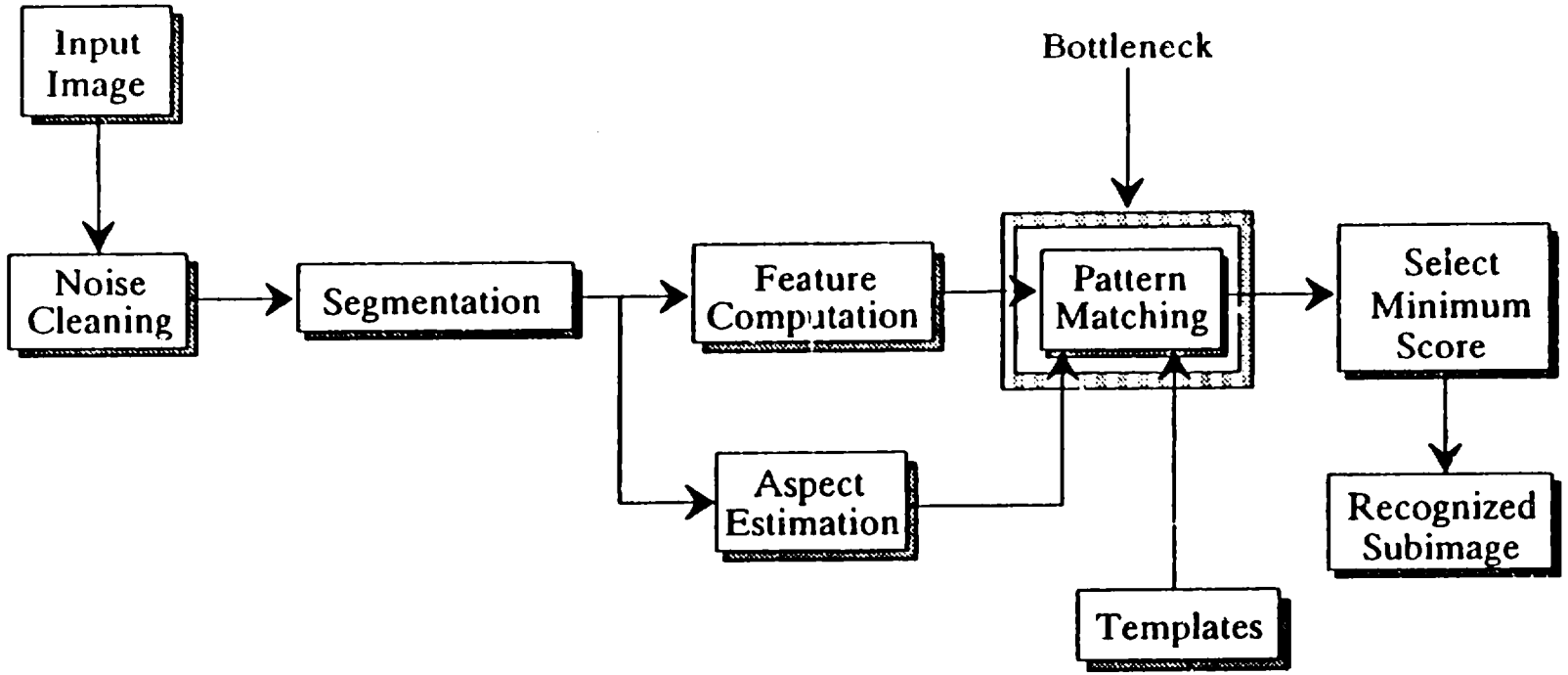


Figure 16. Low-Level Vision Processing - Pattern Matching Approach

wavelength ranges. In such a case, the inputs of the several sensors can combined in the preprocessing step in an attempt to achieve maximum fidelity.

Typically, the first operation after preprocessing is feature extraction. This is usually accomplished by identifying gradients within the image, using the "DOG" (difference of Gaussians) operator. This operator, shown in Figure 17, is created, as its name implies, by subtracting two Gaussian operators to create an operator with two zero crossings. Application of this operator to the image creates a distribution of intensity gradients, and in particular, identifies the edges of objects.

The next step in the processing is termed the primal sketch and first-order segmentation phase. By deciding such things as where the edges of the objects contained in the scene are, which lines and edges at one part of the scene are continuations of lines and edges from other parts of the scene, and what regions of the scene belong to individual objects, a stereotypical "sketch" can be generated from which recognizable features can be extracted. The idea here is to remove the computation from pixel level processing as quickly as possible.

As an example, a tank is represented as moving on a roadway in Figures 18a-d. The "DOG" operator extracts the edges of the tank, which can then be linked during the primal sketch phase. For the tank example, the image has now been segmented into differing regions, corresponding to the turret, the base, the road, and the wheels.

Some method must be devised to recognize actual objects in the primal sketch (for the tank, these are the wheels, cannon, road,...) and relationships among those objects. This is done during the iconic feature extraction and grouping step. Currently, there are several competing ways to do this, and simply stated, they involve intensive, complex numerical and symbolic computations. Symbolic representation of the segmented image occurs at the next level, and finally the resultant symbol set of edges and regions is semantically interpreted in combination with models generated at

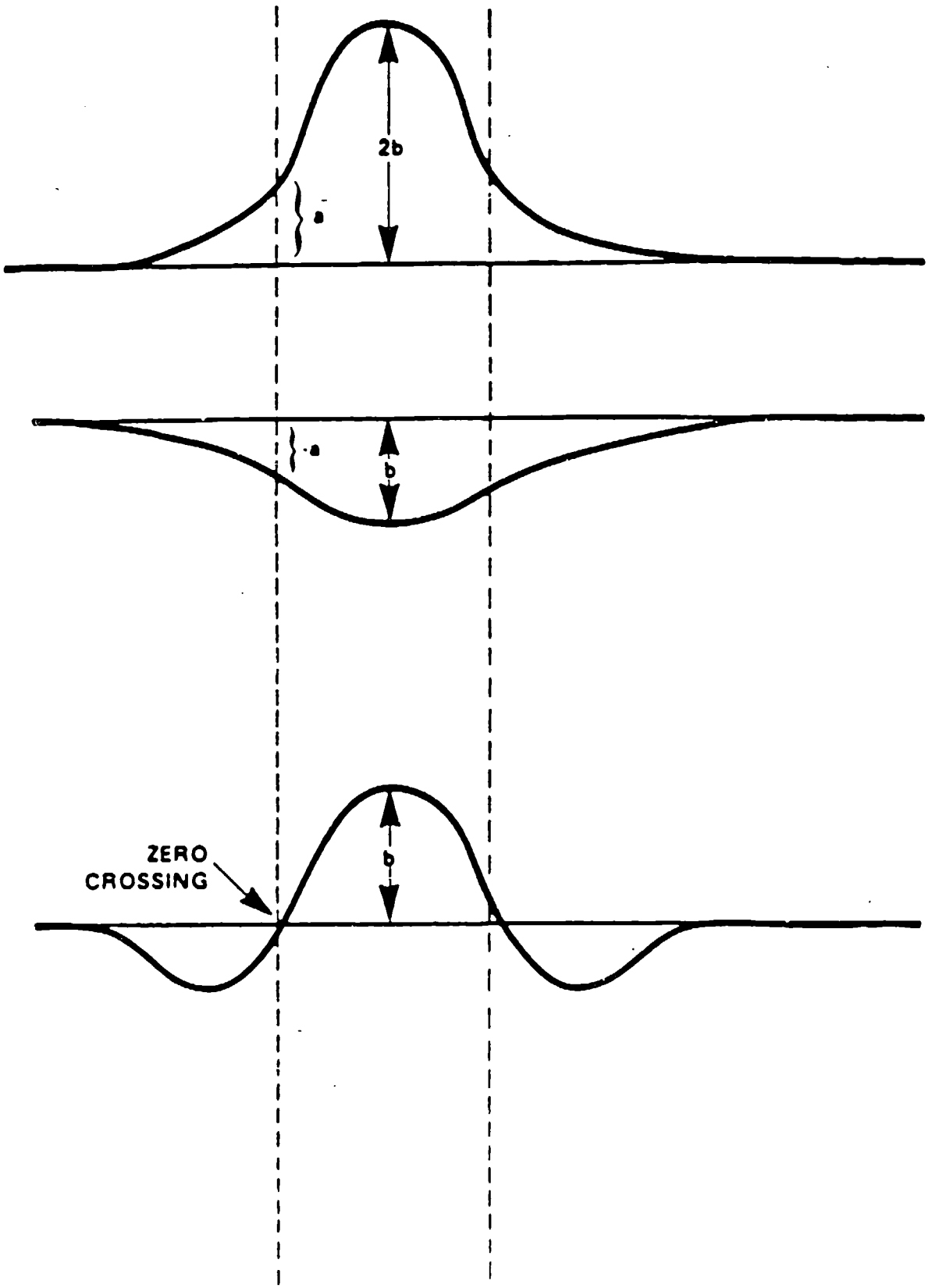


Figure 17. DOG Operator

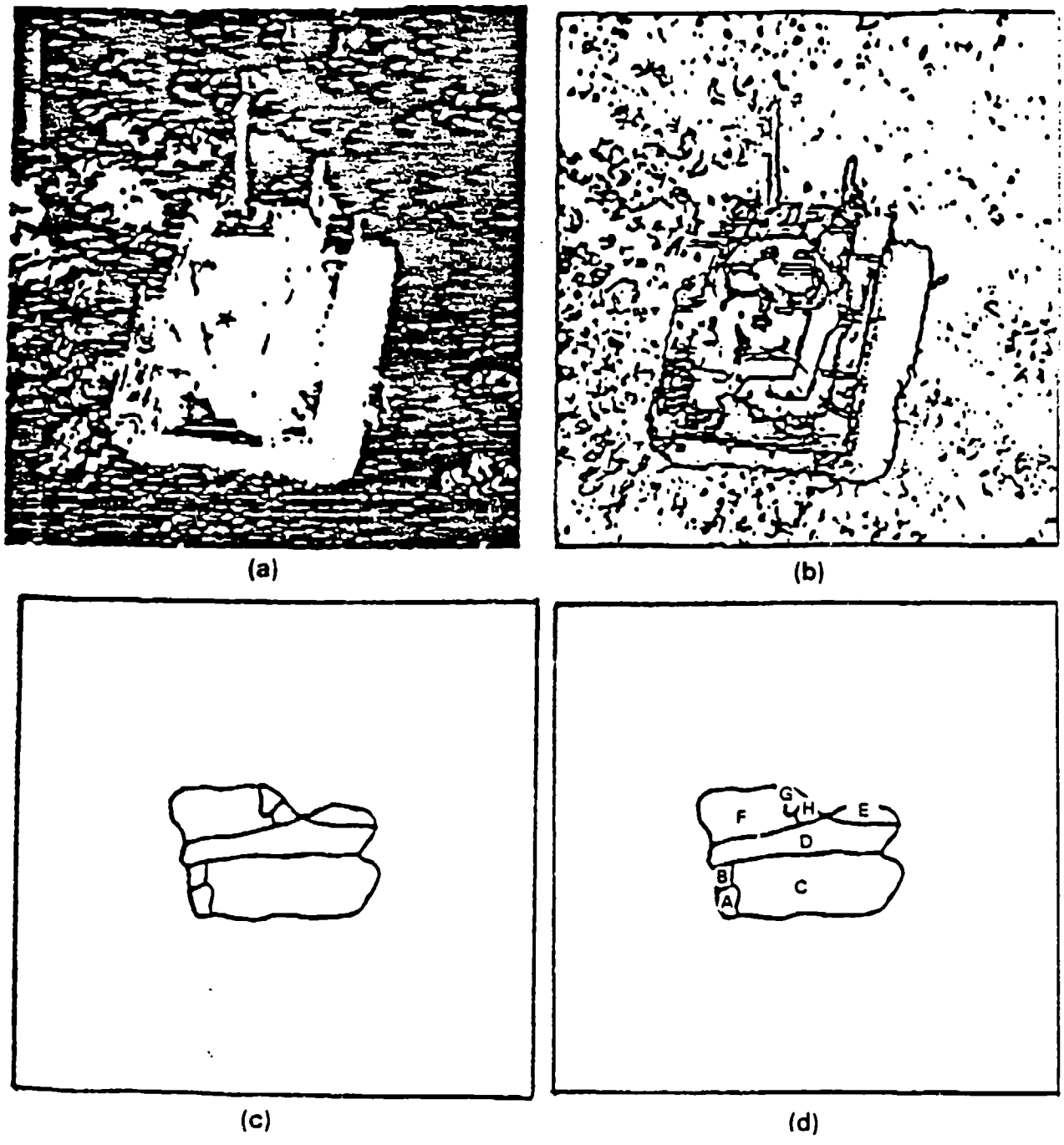


Figure 18. Int-Level Vision Processing - An Example

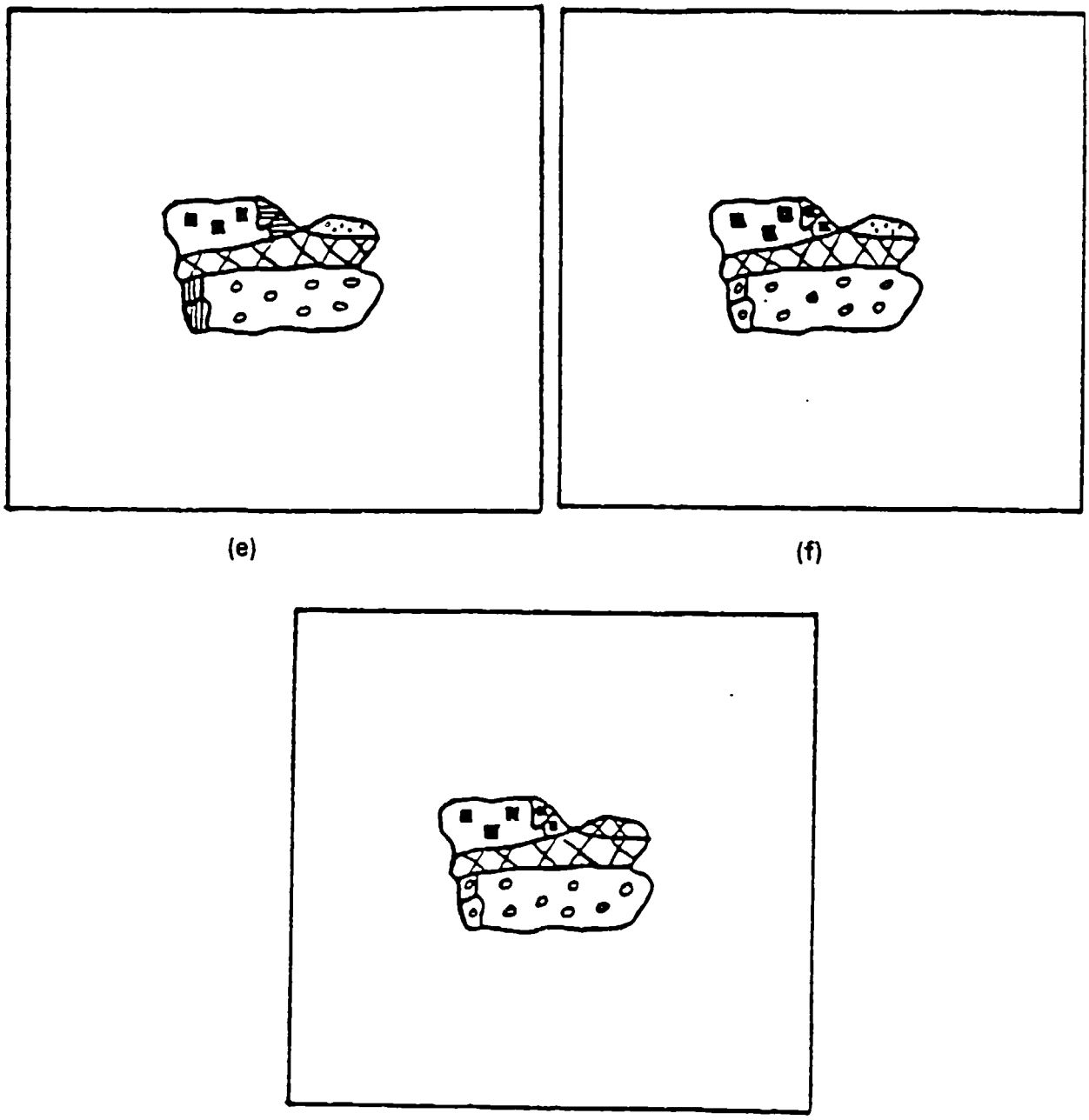


Figure 18. Int-Level Vision Processing - An Example (Continued)

the highest level. This overall system architecture is presented graphically in Figure 19.

As an example of this process, we can look at the problem of identifying that the image is in fact a tank. One aspect of this is determining the identity and relation of the segments from the image in Figure 18. A technique for doing this, based on the use of the "frame" (see Section II.B), is shown in Figure 20. Here, the shape of the turret, and its position relative to the base, match descriptions of similar objects in the systems knowledge base. Within the frame, the turret is seen as deriving from the class gun, and having two specific incarnations: the tank turret and the gunboat turret. Within memory, the frame can call a model of the turret up for symbolic comparison to the segmented image. Then, by recursively repeating this process, the system can potentially identify the tank base. The system can then hypothesize that the segments of the image are elements of a tank. But this hypothesis is not validated until the other elements in the scene have been identified.

As one might expect, this process is a significant bottleneck in machine vision systems, even with the use of domain knowledge to constrain the search space. Multiple hypotheses could be formed, and comparisons made until one inference emerges without contradiction. This offers an opportunity for parallelism in this computation, but at present remains a critical process limiting real time image understanding. However, once object identification is made, the remainder of the computation is handled symbolically.

This leads to the scene understanding phase of the computation, which, unfortunately, is the least understood area of the enterprise. To say that we are looking at an object with a turret or with a top cannon or with tracks is far from understanding that it is a tank as opposed to a tracked troop carrier or some other vehicle. To say that it is a tank is far from understanding what type of tank it is or what it is doing. To do even the simplest of these things will require application of sophisticated modeling methods coupled with concise representations of these objects in the knowledge base of the system. A commonly employed technique uses multiple

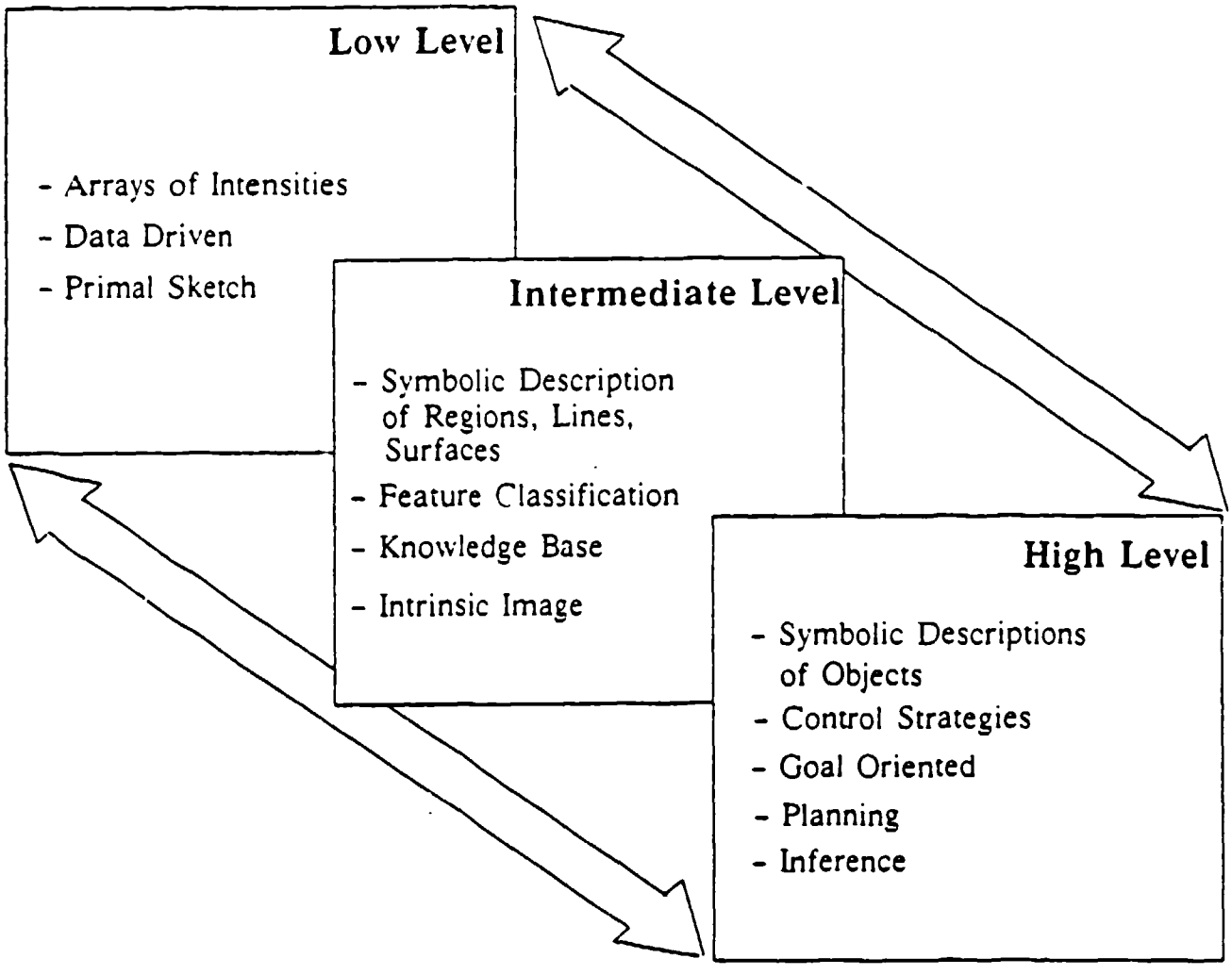
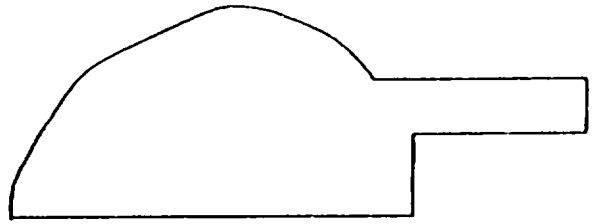


Figure 19. Image Processing Paradigm

Frame:	Turret
From:	Gun
Class1:	Tank
Class2:	Gunboat
Shape:	Oval
Slot1:	Gun
Next.to1:	Tank.Base
Next.to2:	Gunnery.Base
Display:	Model.Turret



Tank Display

Figure 20. Prototypical Frame for a Tank

images to analyze movement or change, and then inferences are made concerning these actions. However, this use of the time domain is not always possible, and, therefore, techniques for scene understanding from context are under development.

In vision processing, knowledge about the scene can be used at all levels to constrain the processing. Initially, geometric inferencing could use knowledge of occluded sides of a three-dimensional geometric object, such as a tank turret, to project the appropriate 2-D model. Domain-specific knowledge will distinguish a road from an airport runway, the difference being the environments around roads versus runways. Finally, identification of the road would increase the likelihood that the scene was that of a tank, rather than a gunboat.

If we had implemented a different control strategy, rather than a bottom-up one, knowledge about the scene would have constrained the system in other ways. As in the case of speech, many types of image understanding control strategies are possible, including the top-down hierarchical approach, mixed top-down and bottom-up, heterarchical approach, and variants of the blackboard approach. In the top-down approach, predictions made from high-level models in the knowledge base are tested and verified, such as the assumption that there is a road in the image. In this case, templates are commonly used to validate higher level hypotheses. In the blackboard approach, the feature extraction, symbolic, and semantic processors operate in parallel and communicate with each other via a common working data storage or "blackboard." The important point here is that knowledge influences all levels of the computation, whatever the control strategy, and very often requires iterative processing at lower levels to test any hypothesis.

From the above discussion, the reader should appreciate that the heart of the vision problem is the recognition of complex objects. Eliminating current computational bottlenecks involves developing new, more efficient algorithms and interfacing techniques to allow flexibility in transitioning between the basic component detection phase, the extraction/feature grouping process, and the high-level interaction with some knowledge base.

The problem is that currently, most vision algorithms run very slowly, and the individual processing stages are not integrated, very often requiring different hardware to achieve the desired functionality. Coupled with this is the need for efficient representations of the knowledge in memory, in a form conducive not only to some variant of template matching, but also to symbolic labelling and context identification. Underlying all of these problems is the drive to develop an efficient architecture for vision/image understanding processing. The belief is that by exploiting parallelism in vision computations, and by developing appropriate implementations of vision algorithms, many of these problems can be overcome. Optical solutions to this problem are also promising, as we will discuss in Section IV.

In fact, the authors believe that vision systems using high level modeling techniques could benefit from optical techniques, since they combine elements of numeric (e.g., edge detection/enhancement for segmentation, Fourier descriptors for feature computation) and symbolic (template matching, object recognition,...) computation.[14] Since such systems are knowledge base intensive, higher memory bandwidth systems, such as those to be discussed in Section IV, may alleviate some of the problems associated with iterative identification processes. Most advanced image understanding systems[15,16] require a smooth transition between numeric manipulations at the pixel and first segmentation levels and the symbolic computations at the higher, object classification and recognition levels.[17] Coupling numerical and symbolic computations in ultimately solving vision problems, such as optical flow, may provide optical computing its most significant role in AI.

In this section we presented an introduction to machine vision, and identified, by way of example, some of the more stringent computational bottlenecks associated with the vision process. At the same time, we drew parallels between the processing associated with images and that conducted with languages, using the speech understanding discussion of the previous section as a reference point. Many of these same issues, such as

hierarchical control and hypothesis testing, will be revisited in the next section, on natural language understanding.

D. NATURAL LANGUAGE UNDERSTANDING

A commonly asked question is:

"What is natural language processing, and how does it differ from speech understanding?"

To answer this question, we will first define natural language understanding, and then explore its relationship with the types of knowledge used in speech understanding. Using an example from a well known piece of optics literature, we will then discuss some of the techniques utilized in natural language processing. This will lead directly into our discussion of expert systems technology in Section III.E.

The most popular definition of natural language, and one that is as accurate as any other, is that it is a language used in spoken or written form as the primary means of communication by a community of people. Hence, in some countries, and within most portions of the US, natural language most often refers to English; in other countries, it is Spanish, Chinese, or one of several other international languages. It should be clear, therefore, that linguistic attributes play a large role in the knowledge processing associated with natural language, and that it is closely allied with the field of computational linguistics.

As was the case with speech and vision, natural language understanding (NLU) had its genesis in man-machine interface research. Here, the desire was to have the machine understand English phrases or sentences input to it through some peripheral device, which was typically a keyboard. At that time, the principal application, in addition to language translation, was the querying of large data bases. The structuring and interpretation of the query was very important, and the user had to be careful that the input was accurately translated into the language of the data base. Drawing upon an

earlier example, a query to find articles in a data base on nonlinear optical materials for 2D Spatial Light Modulators could be structured using natural language input as:

> "Find all articles on nonlinear optical materials for 2-D Spatial Light Modulators."

Early on, the system would have been able to interpret this input only because all words within the vocabulary of the system. Using an appropriate indexing scheme, this input would be transformed to:

> Class: 2-D(w)Spatial(w)Light(w)Modulators
> Subindex: nonlinear(w)optical(w)materials

where the (w) refers to the linking of the words. Even today, many data base query systems still function in this manner. To illustrate the point further, say it was desired to obtain articles on performance parameters of 2D Spatial Light Modulators. The query:

> "Now find all articles on their performance parameters"

would only lead to to an error message from the system, even if this statement directly followed the first. This is because the second phrase contained an indirect reference to an element of the first sentence, so that the subject of the second phrase could not be unambiguously identified. This application of natural language, termed interactive discourse, looks at pragmatic knowledge (see Section III.B) to understand references in conversations, alleviating the problems cited in the example.

In discussing natural language understanding, therefore, we are treating a discipline which is analogous to the human abilities to both read and comprehend texts as well as carry on dialogues. There are no direct parallels between natural language and the signal/image processing common to lower level speech and vision systems. There is a signal to

symbol transformation, but it occurs directly at the input stage, and from that point on, all processing is symbolic in nature. However, one saving point is that much of the same knowledge used in natural language understanding is directly applicable to high-level speech understanding.

In order to get a better appreciation of natural language processing, it is worthwhile to identify some other applications of NLU. While we will address some of the main applications, the interested reader is referred to the book by Rich[18] for others. Aside from input interfaces, a principal application of NLU is in the area of computer programming. Here, the objective is to replace expressions such as:

```
DO 100          I = 1,50
     J = J + DATA (I)
100 Continue
AVE = J/50
```

with:

"Calculate the average of the 50 pieces of data."

Languages are being developed that are more "English-like," especially in the field of AI, as we will see in the discussion on expert systems in the following section.

Another aspect of NLU is text processing, by which we mean the processing of multiple sentences or paragraphs to extract critical information. This specialized extraction of information is very useful in literature analysis, such as the assimilation of information on optical computing from all of the journals of the IEEE and the AIP. Mechanical translation of texts, say from English into Spanish, is another application of text processing. Finally, the generation of natural language output, as in the explanation facilities of expert systems, is another critical application of NLU. The reduction of reasoning processes to a series of simple sentences, however, is a staggering challenge, even for humans.

There are several elements to any natural language processing system. The input is typically derived from a keyboard, although pointing devices or microphones are also sometimes used. In this case, however, the microphone is not responding to words or sentences, but is used instead to input letters in conjunction with a menu-driven software routine.

Following input, the NLU system decomposes or parses the sentence to identify word relationships and dependencies. The parser relies on knowledge about the grammar of the language to identify the subject of the phrase and relate nouns and verbs to their modifiers. For example, the well known phrase from the journal Applied Optics, "Optics is light work," can have a couple of different parsings, examples of which are shown in Figure 21.

Parsing decomposes the input phrase, identifying the relationships between the words and storing them symbolically. Following the parsing, a semantic interpreter takes this information, and, using information from the knowledge base, attaches meaning to each of the words in the phrase. This is accomplished either by lookup, or by conversion to an intermediate format known as the "meaning representation language." This language preserves the meaning and symbolic relationships of the words, but is designed to have a more direct mapping onto the vocabulary of the system than a random input may have. From the above example, the system may determine that either of the following are true:

```
Optics   =   Not_Work (N)
Optics   =   Easy (Adj)      +   Work (N)
Optics   =   Work (N)        on  Light (N).
```

The relative merits of each of these interpretations is determined by higher level processing. In this case, the representation is then passed to the domain and discourse processors, which use pragmatic knowledge to generate hypotheses about the meaning of the input. These hypotheses are either verified by comparison with the knowledge base, or lead to additional semantic, domain, and contextual processing.

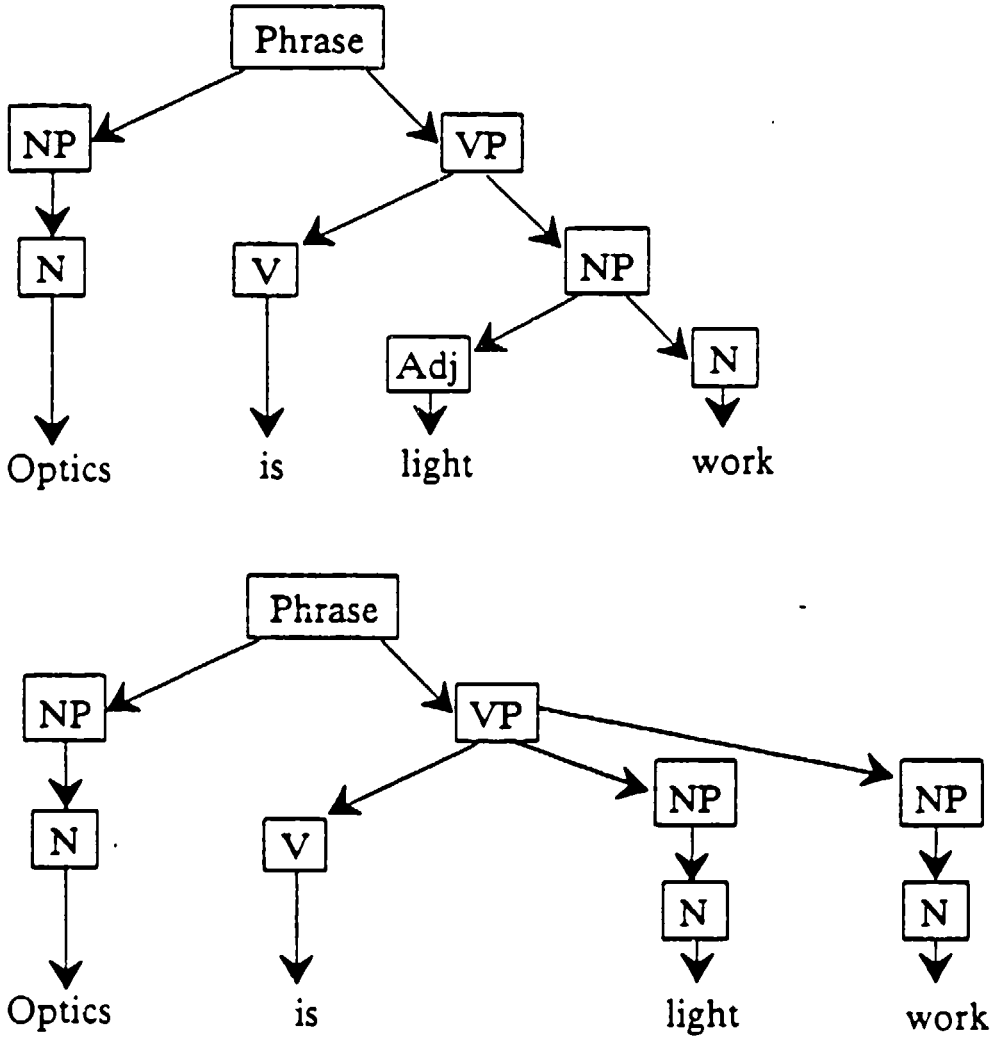


Figure 21. Possible Parsings of the Phrase " Optics is light work"

In the above example, a bottom-up paradigm was used to explain the knowledge processing, just as we applied it in the cases of speech understanding and vision. The bottom-up approach, shown graphically in Figure 22, can be used to move the processing up from the parser to the semantic interpreter, domain and discourse processor, response planner, and data and knowledge base translators. However, other approaches are equally plausible. For example, a system architecture based on the blackboard model (Figure 23) may be useful to combine partial understandings by the processors and achieve full understanding more rapidly and in parallel. For example, a parser may find two different parsings of the sentence "Optics is light work," but other processors will have to determine the most likely interpretation by using other knowledge.

Unfortunately, natural language processing is at present unable to unambiguously determine the meaning of input sentences, or use contextual or pragmatic knowledge effectively. Each phase of the processing is a computational bottleneck in its own right, particularly when dealing with input information which is flawed (e.g., wrong punctuation), or has errors (e.g., misspellings). In the understanding of raw text, natural language systems need to be more broadly applicable, more robust. They are currently slow and limited to understanding text only in very narrow areas, with a low accuracy of interpretation. As was the case with speech and vision, there has been a tradeoff between generality and performance in natural language processing systems.

An overall goal of natural language research is the development of a sufficiently robust system (necessarily domain portable) which can achieve high levels of accuracy in interpretation. As before, the desire to optimize performance has initiated the development of parallel algorithm research, with an eye towards achieving optical or multiprocessor implementations. But, as in the case of vision, our understanding of parallel tasking in not sufficiently developed to understand the implications of this research.

From the preceding discussion, we can begin to see how a multiprocessor or optical processor could be used to achieve understanding of

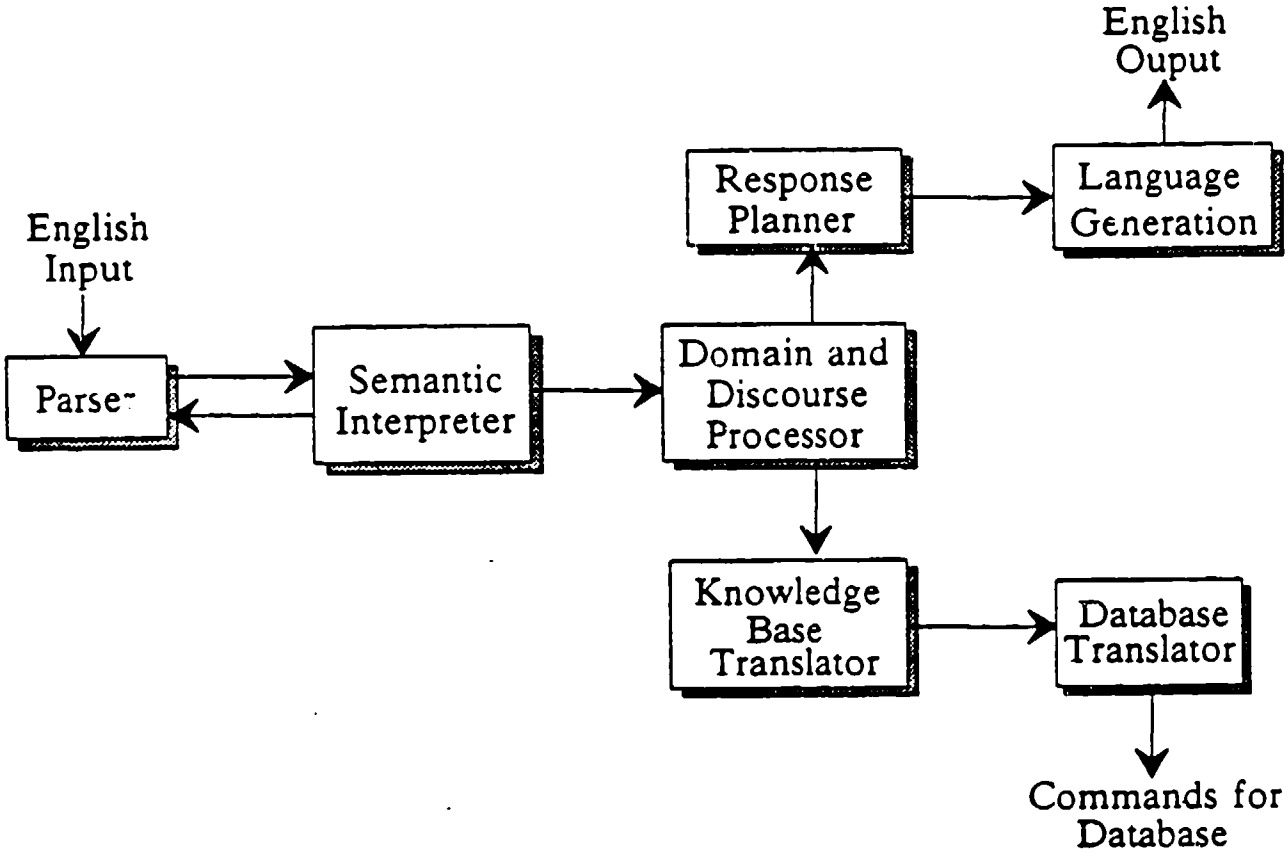


Figure 22. Elements of a Natural Language System

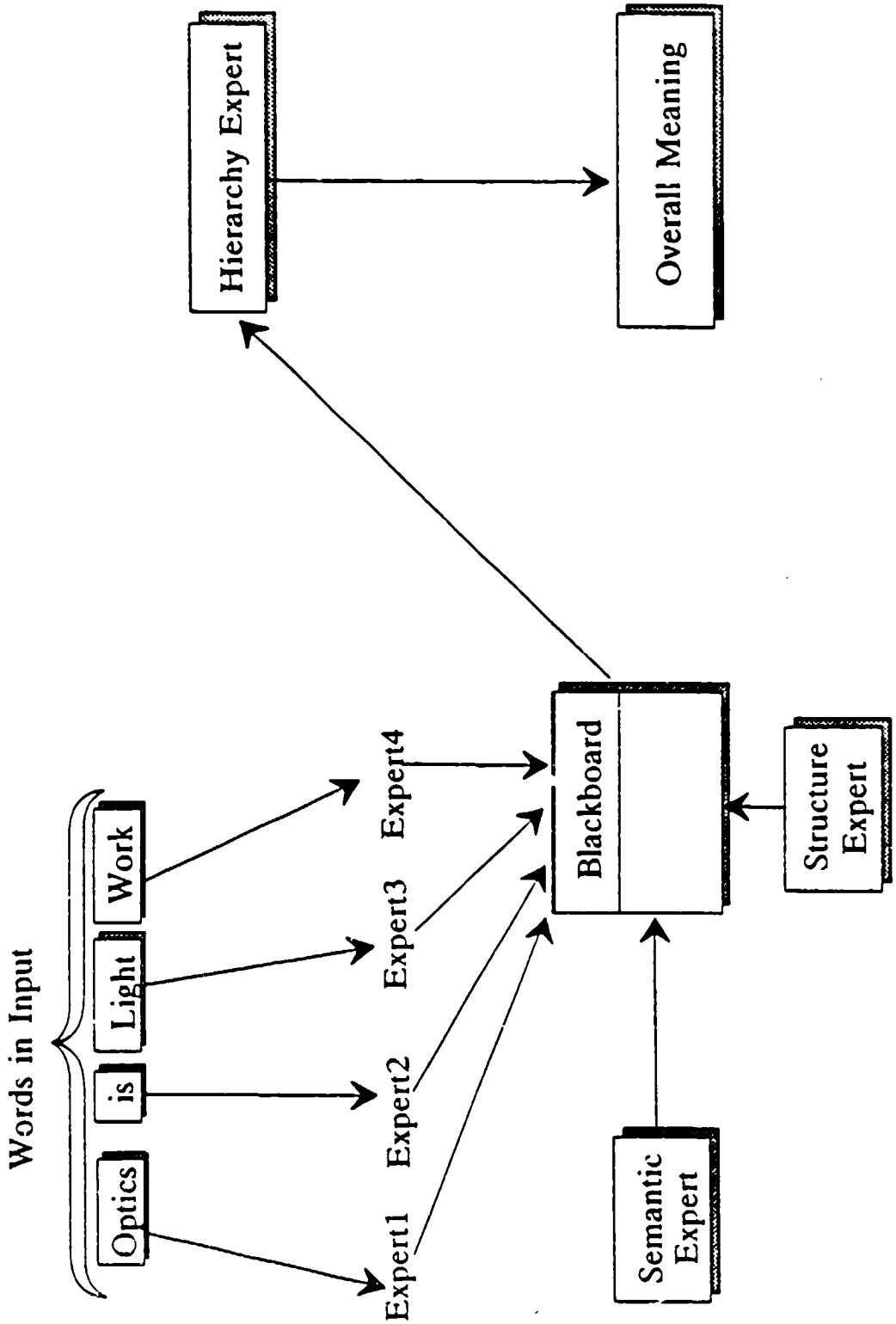


Figure 23. Natural Language, a Blackboard Model

natural language in real time. A parser itself could be parallelized, since many different morphemes, lexical variations, and syntactical structures can be analyzed simultaneously. A blackboard approach to the main processors would be immediately parallelizable.

The final point to be made is that, in describing natural language processing, we are in fact discussing a behavior of computer systems. We have crossed into the area where we are looking at systems that can interpret and reason about inputs, hopefully even learn from their mistakes. The issue of interpretation expands into the problem of where does natural language processing end and where does expert system processing begin? Knowledge representation becomes more and more common to both, since stored knowledge will have to be used in order to understand new natural language input. General knowledge about the world, i.e., common sense, as well as specialized knowledge about a problem domain must be used to determine if the new input is literally plausible and corresponds to physical reality, or if it is erroneous as opposed to metaphorical, humorous, or sarcastic. These questions will surface again in the next section, where we will look at the discipline known as expert systems.

E. EXPERT SYSTEMS

The final capability of symbolic computing that we will discuss is expert systems. These systems are characterized by their almost total reliance upon manipulations of symbolic information, in marked contrast to the systems presented previously. In speech, vision, and natural language understanding, the emphasis was on some form of signal-to-symbol transformation, coupled with the use of high-level knowledge. These are disciplines which are dependent upon knowledge retrieval and reasoning processes. Expert systems, on the other hand, involve the process of knowledge acquisition in addition to the retrieval and reasoning process. Advances in the other functional capabilities would be of great benefit to

expert systems technology, since it draws heavily on the interfaces provided by these disciplines.

But what is an expert system? In its broadest sense, an expert system is a machine which mimics or emulates the thought and reasoning processes of a human expert. It seeks to utilize the solution techniques utilized by the human expert to solve a particular problem. These techniques are, in many cases, just the rules of thumb or heuristics described in Section II. The way experts look at a particular problem, the information they look at, the data they require, their knowledge about the knowledge they possess (what we term metaknowledge), what they ignore - these are the elements we call expertise. Expert systems attempt to capture this expertise, and apply it to a particular problem area or domain. The price to be paid, as we have seen in earlier sections, is one of generality; to date, expert systems have only been able to function in very specialized, narrow areas of expertise. Some of these successes were cited earlier in this section: R1,[8] the system that configures VAX and PDP series minicomputers; DENDRAL,[7] the system developed to interpret spectroscopic data; and MYCIN,[9] the system which aids physicians in making diagnoses in internal medicine.

In each of these systems, knowledge was placed into the machine which enabled it to "understand" the problem, in much the same way as was done for speech, vision, and natural language systems. It was then possible for the machine to function as a decision aid to the user. The system used knowledge retrieval and reasoning to be expert, generating inferences about the problem based on data supplied by the user. The power of these expert systems is the amount of symbolic information which they can store in their knowledge base, and their ability to rapidly process it.

There are three components to any expert system (see Figure 24) - the knowledge base, the inference engine, and the explanation facility. This expertise is stored in the knowledge base, using one or more of the representation types described in Section II.B The knowledge base includes the heuristics as well as any elements of metaknowledge required for the task. The inference engine gives the expert system its reasoning capability, allowing the system to combine rules or frames to generate a conclusion.

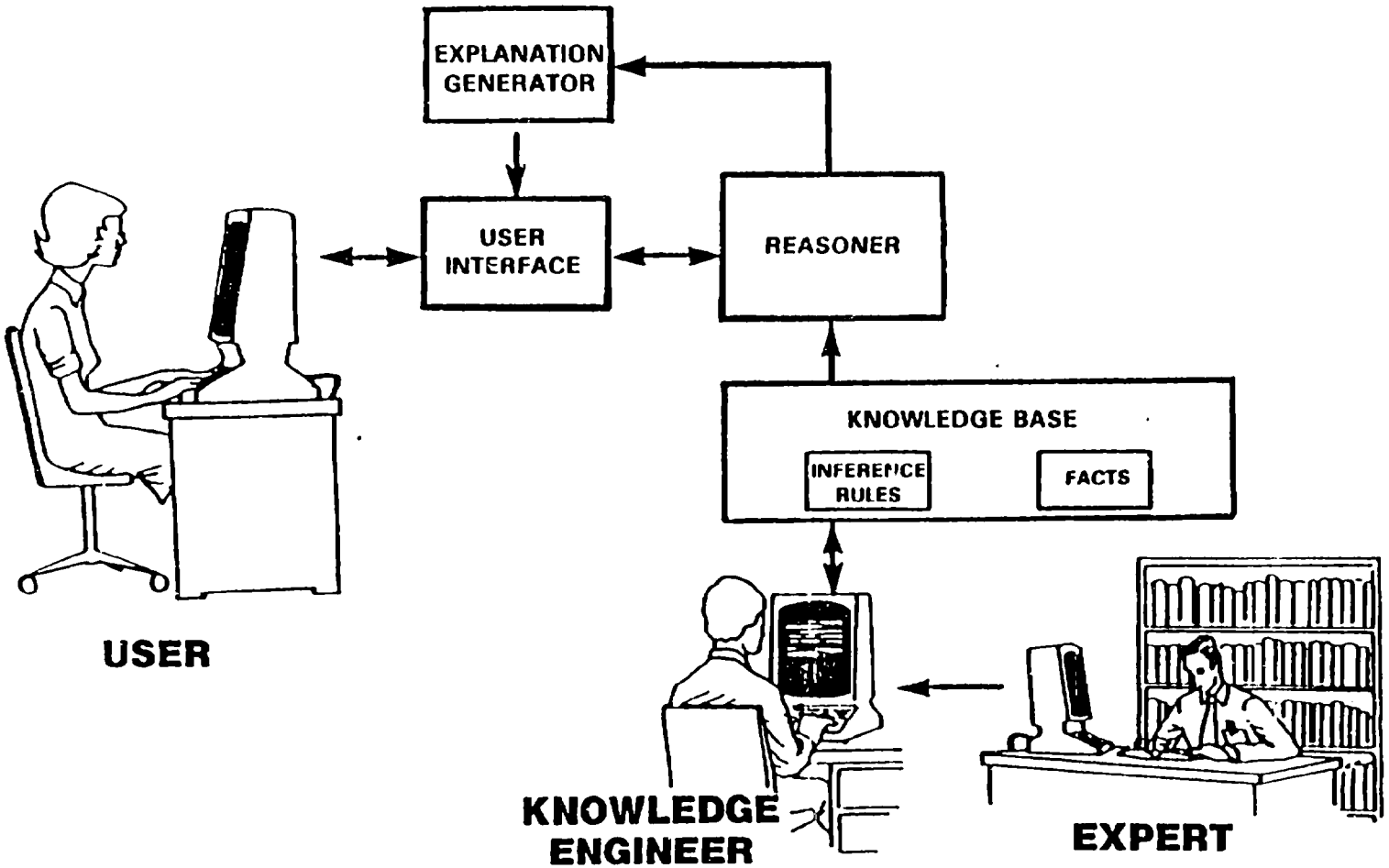


Figure 24. The Elements of an Expert System

This usually includes backward chaining, or goal-directed reasoning, as well as forward chaining and expectation-driven reasoning. The algorithms that underlie these reasoning processes are very often matching procedures, comparing the truth of one statement relative to that of another. As an example, the Rete algorithm, used in the popular OPS-based expert systems (such as the R1 system cited above), matches antecedents of production rules to the global memory in conducting a backward chaining operation.

The explanation facility is what allows the user to understand the machine's solution strategy, to determine why particular conclusions were reached. Very often, the explanation facility is a modified natural language interface, allowing the user to ask the machine:

"How did you arrive at this conclusion?"

and receive an answer which shows the evolution of the machine's reasoning process. For a rule-based system, an appropriate response to the above question could be a list of rules that were "fired," much like the trace of a FORTRAN program. As the size of the knowledge base increases, and the expected number of rule firings scales proportionately, this explanation technique becomes insufficient. What is required is a system that can summarize the reasoning process employed by the system. Such an interface could take advantage of research in natural language understanding, seeking to allow those systems to cooperate with the expert system, possibly in a blackboard architecture.

The development of a "classical" expert system typically requires a team of three people - the expert, the knowledge engineer, and a symbolic programmer. As we stated before, the expert supplies the knowledge and expertise to the system, which is stored in the systems knowledge base. The knowledge engineer has the task of acquiring the knowledge from the expert, typically via interviews and by presenting the expert with simulations of the problem. This is programmed knowledge acquisition, and is independent of any knowledge obtained from external sensors or learned by the system itself. The symbolic programmer takes the input from the

knowledge engineer and literally programs the knowledge into the system. Since this is a labor intensive problem, early expert systems, such as the ones cited earlier, required many man-years of effort to develop.

This division of labor has been eased greatly by the advent of advanced expert system building environments, and now, the knowledge engineer and the symbolic programmer are very often the same person. These building environments are "shell programs," containing all of the tools that a programmer needs to develop an expert system. In such a tool, an organized knowledge base is supplied; but it is devoid of any knowledge. The inferencing ability is also supplied, allowing conclusions to be generated once the knowledge base is "filled." Drawing an analogy with spreadsheet programs for microcomputers, these environments provide the equivalent of an empty spreadsheet. The programmer has the ability to input knowledge into the knowledge base, much the same way that an accountant can input figures into the spreadsheet, and tailor the program to meet his or her needs. Finally, the ability to combine heuristics to reach conclusions is analogous to the combination of spreadsheet cells to form a new entry. Just as the spreadsheet revolutionized the use of microcomputers, these building tools have decreased the development time associated with expert systems from several man-years to several man-months, depending on the level of difficulty of the problem.

The knowledge base of an expert system is its power, since it has been found that specialized knowledge is an essential adjunct to logic. The arduous process of incorporating that knowledge into the machine is a major limiting factor, even in narrow domains. Interactive knowledge acquisition tools, such as TEIRESIAS,[19] have proven their usefulness in helping the domain expert express knowledge in forms compatible with the knowledge base. More sophisticated systems can be built to infer rules themselves from presented data, as was done with MetaDendral.[20] Even better, we can build systems which can learn to guide their own search strategies, i.e., learn heuristics, as was done with ACT[22] and EURISKO.[21] But the principals of learning and adaptation are still poorly understood. Consequently, most expert systems are not currently learning systems. Rather, they use the

application of prestored knowledge instead of learning from problem solving, from failures, and from correcting errors. The goal of developing systems that learn, in the form discussed in Section II, could greatly decrease the time and effort spent in the development of knowledge based systems.

An advanced feature of intelligent systems, and expert systems in particular, is their ability to pursue multiple lines of reasoning in generating a conclusion or decision. This has the advantage of being able to prune or reduce the potential solution space by applications of either goal directed (backward chained), model directed, or forward chained reasoning. This is a very powerful technique, which, although still in its infancy, will allow a program to be approached from multiple angles until a solution is generated. But any opportunity for conducting simultaneous, multiple viewpoint reasoning is heavily dependent upon developing the appropriate parallel computational structures. This is due to the increased amount of knowledge based interaction in such a system, which would only worsen an existing bottleneck in uniprocessor-based AI systems. Unfortunately, as we will see in the next section, our understanding of parallelism in computations is rather limited. But the potential benefit of parallelism in symbolic computation is one factor which makes optical computing particularly attractive.

Having clarified what an expert system is, we can now investigate what they can do. Hayes-Roth, Waterman, and Lenat, in their guide to expert systems,[10] identified ten areas of application for these systems. These ten, shown in Figure 25, suggest that there are a large number of potential applications of expert systems, for everything from decision aids, to the construction of a laser (within given constraints), to isolating the source of a failure in an optical system.

We would like to focus on the last of the above instances in a simplistic example of application (3). We have taken some liberties in developing the structure of this example, and we realize that it deviates somewhat from standard experimental practice. Nevertheless, we feel it

(1) Interpretation of sensor data;
(2) Prediction of consequences in given situations;
(3) Diagnosis of malfunctions from observables;
(4) Design of objects within given constraints;
(5) Planning actions;
(6) Monitoring or comparing observations to plan vulnerabilities;
(7) Debugging and prescribing remedies for malfunctions;
(8) Repair or execution of a plan to administer a prescribed remedy;
(9) Control of systems; and
(10) Instruction, which includes diagnosing and remedying student behavior.

Figure 25. Applications of Expert Systems

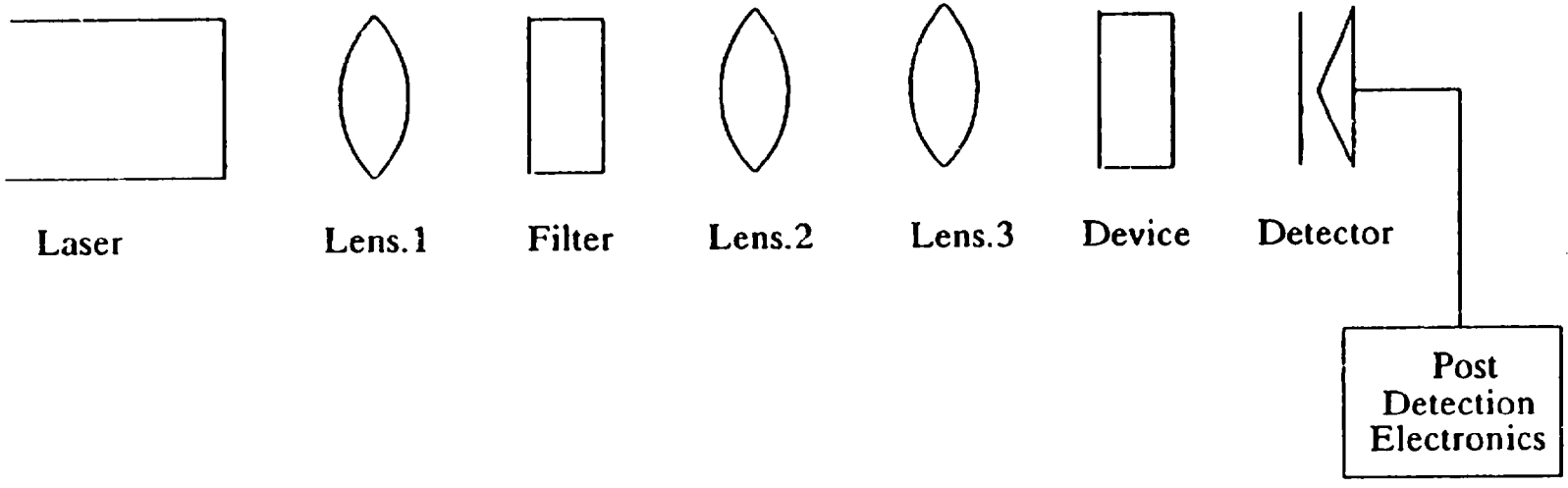


Figure 26. Prototype Otpical System Used in Expert System Example

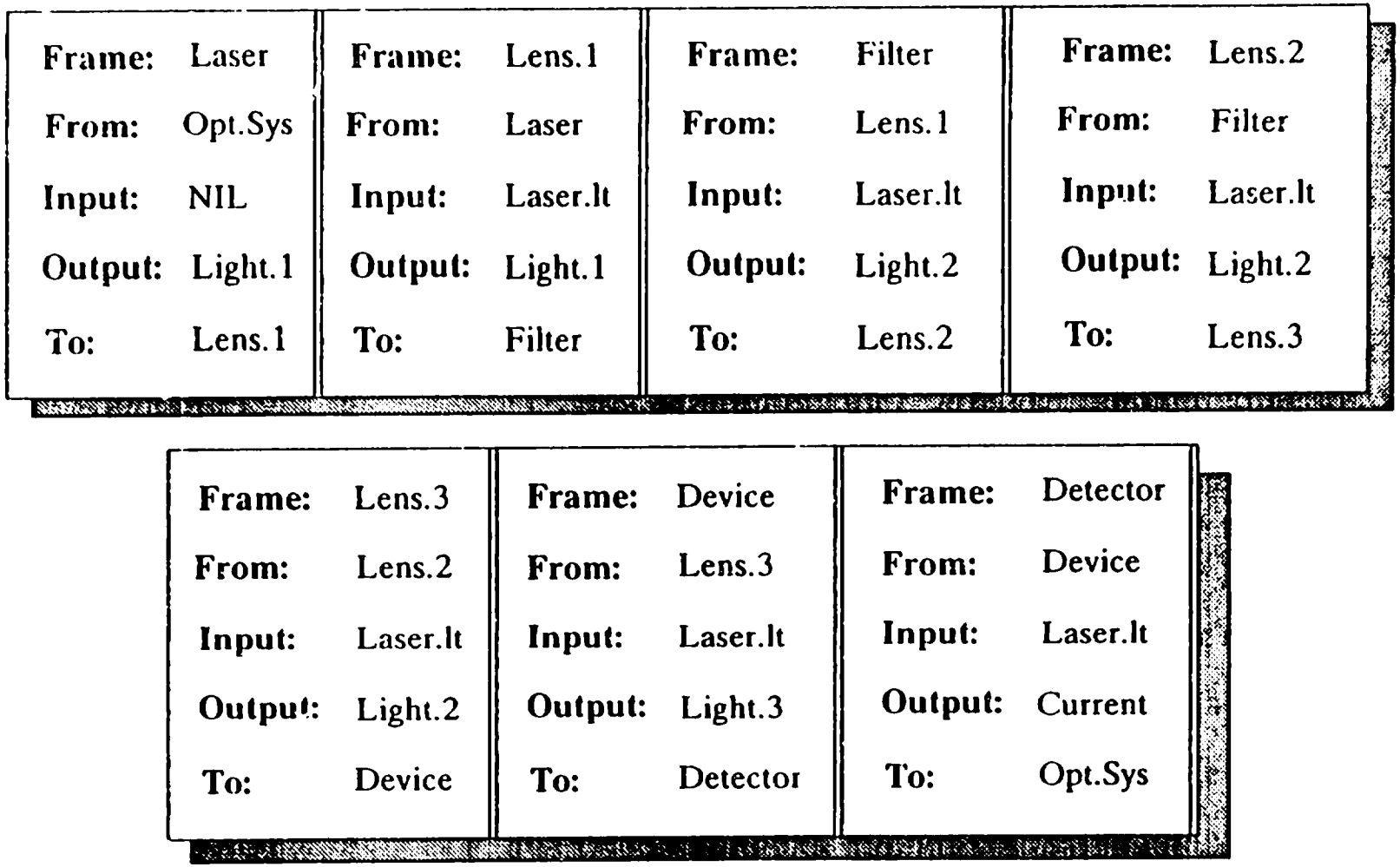


Figure 27. Frame-Based Representation of Optical System in Figure 26

will be useful in conveying the principles of expert systems, as will also serve as a summary of the ideas discussed in previous sections.

For the purposes of this example, let us consider a system which consists of a laser, a narrow spectrum bandpass filter, a nonlinear optical device, a detector and a couple of lenses in the configuration.

In this system, the laser emits light into the first lens and subsequently into the filter, where the beam is changed or modified in some way. In this example, we will assume that the filter only passes light corresponding to the exact wavelength of the laser, so that any variations in the laser's output spectrum will lead to a decrease in light beyond the filter. This "filtered" beam is then transmitted through some nonlinear optical device, and the output of the device falls on the detector. Using a series of frames to represent this system, at the most simplistic level we have the following elements in the knowledge base of the system:

We have represented the optical path of the system by the slots **From:** and **To:**, reminding us of a semantic net relationship between the various frames. The changes to the light as it moves through the system are represented by the variations in the slot **Output:**. Finally, the attribute **Input:** stores the knowledge that laser light moves through the system. Let us now suppose that the detector's output drops to zero ((= Current NIL), in LISP code), indicating a problem, and the system needs to assist the user in determining the source of the problem. In this context, our simple expert system can function as a diagnosis aid.

In our example, the system could proceed in one of two ways, either working backward from the detector or forward from the laser. In the former case, the system could hypothesize that there is no output because there is no input; in other words, there is no laser light reaching the detector. For this to be true, one of the following must also be true: the detector is faulty, the nonlinear device is faulty, or a component

earlier in the optical path is the problem. Here is where some expertise can enter the problem, since the system may know that:

Rule.1: If the detector output drops to null, then the detector is not at fault.

Rule.2: If there is a nonlinear optical device, then the outputs of the device are sensitive to alignment.

Applying this expertise to the problem, the system can eliminates the detector and other components as sources of the problem, and hypothesizes that the nonlinear device is at fault. By interacting with the user to obtain the additional required information, the system can resolve the truth of the initial hypotheses. This interaction with the user may take the form of:

System: "Is Device input still equal to Laser.lt?

User: "Yes."

With this additional information, the system then concludes that the problem is in fact with the device.

In the other paradigm, the system could assume that the laser is faulty, and work its way out to the detector by way of:

If no laser.lt from the Laser, then no laser.lt into Lens.1; Lens.1: Input = Null;

If Input = Null, then Output = Null;

If no laser.lt from Lens.1, then no laser.lt into Filter; etc...

finally concluding:

If no laser.lt from the Laser, then no laser.lt into Detector;

If Detector Input = Null, then Detector Output = Null.

Having generated the hypothesis that the laser is defective, it could then verify the hypothesis by asking the user:

System: Is the output of the Laser still equal to Laser.lt?
(Does Laser: Output = Laser.lt?)

If the answer is positive, some other component in the optical path is defective. The system can then hypothesize that another component is faulty, and iterate on the above process.

The above example is typical of expert system operation. The system reaches conclusions based on the truth of a series of knowledge base elements at any particular point in time. Hypotheses are generated and validated through additional interactions with the user, who supplies the necessary information about the state of the system. In this manner, expert systems can function as a diagnostic aid to a variety of users.

The example may also allow the reader to appreciate the difficulties associated with incorporating knowledge and expertise in AI systems. This is another instance of our recurring theme of the tradeoff between generality and performance. One of the limitations of expert systems is the narrowness of the domain of expertise incorporated into a system. Each system is a relatively isolated project, and, as a result, the techniques developed to solve the particular problem are not applicable to all expert systems. And increasing the size of the knowledge base, equivalent to expanding the domain of expertise, just leads to a combinatorial explosion of possible inferencing and machine states. There are also hardware limitations, such as the size of the working memory in the computer, which can only store so much knowledge at any point in time.

The reader may ask why increasing the size of working memory, thus enlarging the active knowledge base at any one time, will not at least partially alleviate the narrowness problem. It does, but we are still faced with the difficulty of organizing and managing the knowledge, and then channeling it through in serial fashion to the processor. This is complicated by the difficulties in representing certain types of knowledge, such as time-varying data, data with certainty which varies over time, and knowledge about processes and causality. The representations discussed in Section II.B are inadequate for many tasks because they are unable to appropriately store the time variations or statistical uncertainties associated with that knowledge. Going back to our example on isolating a failure in an optical system, use of the words usually and typically in the rules imply an uncertainty in the knowledge:

> Rule.1: If the detector output drops to null, then the detector is usually not at fault.
>
> Rule.2: If there is a nonlinear optical device, then the outputs of the device are typically sensitive to alignment.

At present, our abilities to represent these types of knowledge limit the breadth and robustness of most knowledge-based systems.

As an example, consider the problems associated with the representation of objects in space and the spatial and temporal relationships among them. Such problems, which are commonplace in vision research, also arise in representing relationships in expert systems. How will a computer "understand" that both the light which is incident on a lens and the light that emerges from the other side are really part of the same beam? Other types of knowledge may best be stored non-verbally, such as in visual images. Afterall, a picture or graphic may be worth many line of computer code. But the appropriate way to store graphical knowledge so that it can be updated, retrieved and used in making inferences is an unsolved problem.

Even if we could expand the domain of expertise, computational bottlenecks exist in the knowledge processing which limit the performance of most expert systems to between 10- and 1000-rule inferences per second (RIPS). For purposes of comparison, this corresponds roughly to throughputs of 1 to 100 MegaFLOPS for numerical computations. This is not purely a hardware bottleneck, since much of the computational load rests in how the software structures the knowledge retrieval and reasoning processes. As an example, one goal of expert systems research is to effectively modularize the way knowledge is stored and manipulated. At the present time, there is virtually no separation or modularity between the various components in an expert system - the explanation facilities and user interfaces share the same memory with the knowledge base and the inference engine. Organizing, tracking, and processing all of this knowledge in uniprocessors effectively limit the rate at which symbolic computations can be performed. This is an example of the famous "Von-Neumann bottleneck," which will be discussed in the next section.

These difficulties currently limit the size of the knowledge bases and hence the overall robustness of the system. What is desired is shown in Figure 28, where the components of the expert system have been modularized, but are still interacting at multiple levels within the system. Each component could possibly function on an independent processor, with each parallel process communicating by sending messages to other processing elements. This separation could allow for expert systems with larger domains, more robust inference capabilities, and more diverse applications.

F. CONCLUSION

Having generally introduced the main functional capabilities within symbolic computing, we have seen that each focuses in on improving the machine understanding of the domain in question. Understanding, in a limited sense, is achieved through extensive interactions with the knowledge base of the system. The main characteristics of intelligent systems, the heuristic retrieval of knowledge and the various reasoning

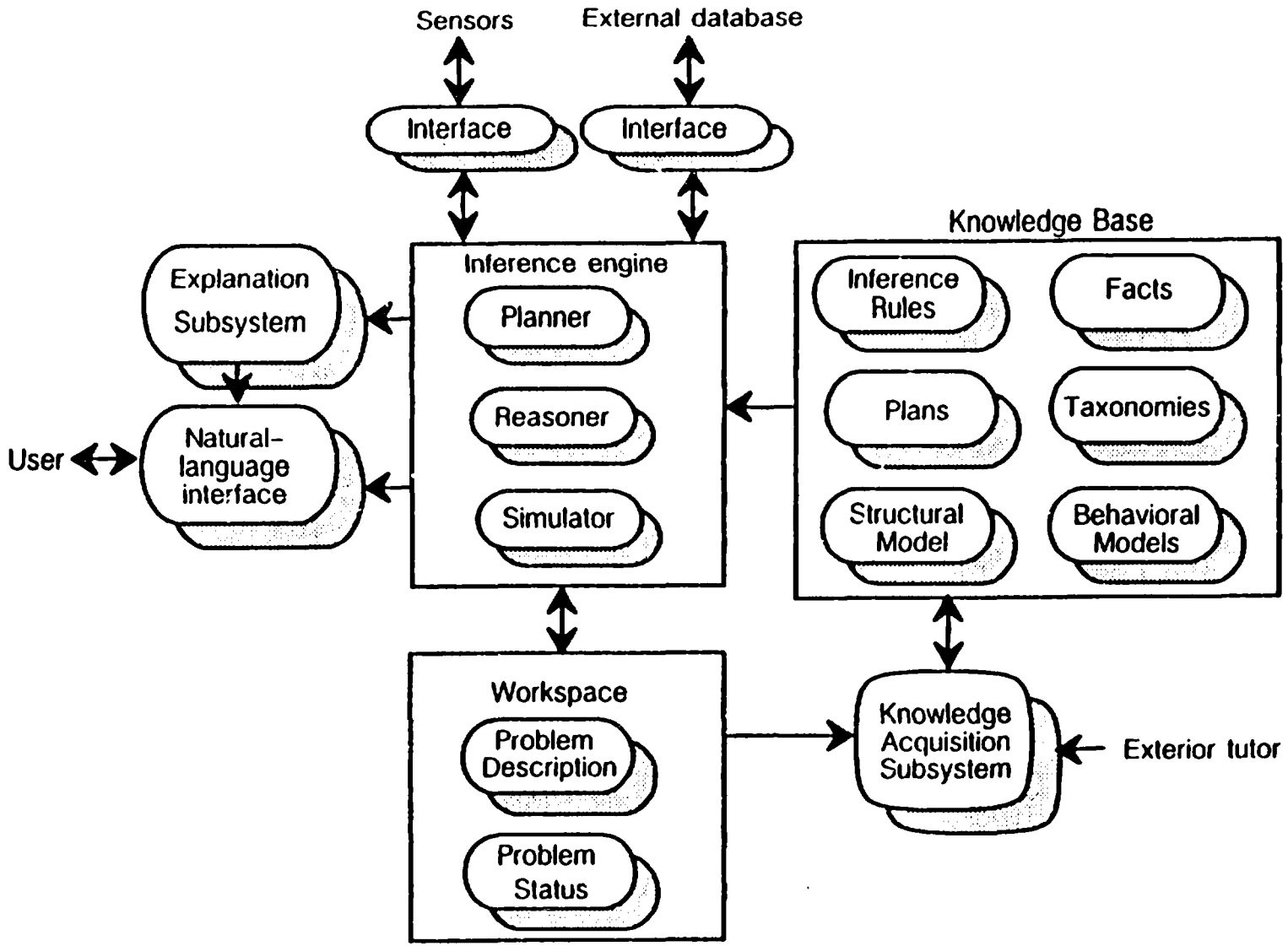


Figure 28. A Modular, Parallel Expert System

processes, were central to each of the capabilities discussed in the section. The acquisition of knowledge, and subsequent programming into the system, played a leading role only in the expert systems area. However, relative to current system performance, it is this organization, manipulation and processing of knowledge that present the main computational bottlenecks in AI systems.

Another point worth noting at this juncture is that the types of operations performed in AI systems are, for the most part, very domain specific. In many cases, the actual instructions, memory utilization techniques, evaluation and matching metrics, and search processes are embedded within the control sequence for the intelligent system. It is therefore very difficult to cite common operations, other than to show the overlap of techniques in high level processing. An example of this case be seen in the area of expert system development tools, where each tool possesses its own control structure, its own group of representations (production rules, semantic nets, frames, scripts, etc.), its own reasoning paradigm (forward chaining, backward chaining, expectation driven, or a combination of these), and, as a result, its own matching or evaluator structure. While any of these tools may have some operations in common with others, typically the variations are quite large from environment to environment.

An underlying theme of this section was the difficulty associated with representing knowledge in AI systems. To be useful, and reasonably general, multiple representation schemes will likely have to be used within any given system to express various kinds of knowledge, especially the difficult areas mentioned above. A problem which has quantities of both declarative and procedural knowledge apparently needs multiple representative schemes. Thus, systems will evolve with ever larger knowledge bases and with multiple types of representations, in order to address more sophisticated and more general problems. The challenge will lie in the the organization and management of this knowledge so that it does not lead to additional bottlenecks or decreases in the processing rate.

It has been estimated that just to overcome existing AI system bottlenecks, the computational throughput rates of symbolic computers will have to be increased by several orders of magnitude over current capabilities.[23] While highly parallel systems are being studied to address this problem, limitations in our current understanding will make it evident that parallelism alone cannot achieve the required speedup in computation rates. Use of alternative systems, such as optical computing systems, show great promise if the appropriate coding schemas and operations can be made optically compatible.

The previous sections have highlighted topics of current interest in symbolic computation, with an eye towards identifying potential applications of optical techniques. Applications such as vision and speech recognition have direct mappings into image and signal processing, tasks which optical processing and computing techniques are particularly well suited to. The major challenge for optics is the application of knowledge processing techniques on top of this lower level processing.

The main point in the earlier discussions was that the operations where optics excels are, for the most part, the same types of operations utilized in symbolic computations. So, while the promise of applying optical techniques to AI problems exists, some strong challenges remain, such as optically compatible representations, understanding parallelism (in optics and in computations in general), memory interactions, proper optical/electronic interfaces, and the ability to implement the desired architectures and required component technologies. These ideas will be explored in greater detail in the next section, which looks at potential architectures for symbolic computation.

CHAPTER IV
SYMBOLIC COMPUTING ARCHITECTURES

A. INTRODUCTION TO SYMBOLIC ARCHITECTURES

The previous Sections have laid the groundwork for the remaining discussion of symbolic computer architectures. Reflection on our earlier description of the fundamental characteristics of symbolic computing should raise an awareness to the importance of relationships between data elements - the relationships between objects and attributes in LISP, the relationships between nodes of semantic networks, the relationships within and between frames, etc. This has led computer scientists to investigate the performance improvements that could be forthcoming by the use of computer architectures for which the connectivity between processor nodes could reflect the relationships fundamental to symbolic computing. Such thinking derives partially from experience with numeric computers in enhancing performance by matching architecture to algorithmic structures and visa versa. Of course, one must keep in mind the importance of maintaining flexible architectures that can adapt to changing relationship patterns; otherwise, the result will be special purpose machines with limited utility.

The similarity of these highly connected architectures to neurological systems lends credence to their importance in symbolic processing. The brain, with its relatively slow components (neuron speeds on the order of milliseconds), is able to process symbolic information at rates several orders of magnitude faster than conventional (von Neumann) computer architectures. Two differences between the electronic and biological systems that stand out are the connectivity between components and the intermix between processor and memory operations. Neurons in the brain can have upwards of 10,000 synapses (biological connectors) whereas their electronic counterparts, gates, typically have only a few connections to other gates. In the area of memory distribution, the von Neumann architectures are characterized by a separation between the processor and

memory functions; and the transfer of information between the two often results in a speed bottleneck.

A criticism of von Neumann architectures is in no way intended. Afterall, when von Neumann proposed this processor/memory separation, it was based on the limitations imposed by the technology of that time period (late 1940s). Also, such architectures have proven vastly superior to the brain in performing numeric operations. Although it is beyond the scope of this Chapter to discuss optimal symbolic/numeric systems, future systems may someday consist of cooperating symbolic and conventional numeric processors.

Computers with high levels of connectivity and distributed memory have been labeled parallel processors due to their capability of supporting concurrent operations. Before discussing the potential for optical parallel processing, we would like to familiarize the reader with the principles and terminology of parallel processing in general, and to discuss broad categories of parallel architectures.

B. ARCHITECTURES FOR PARALLEL PROCESSING

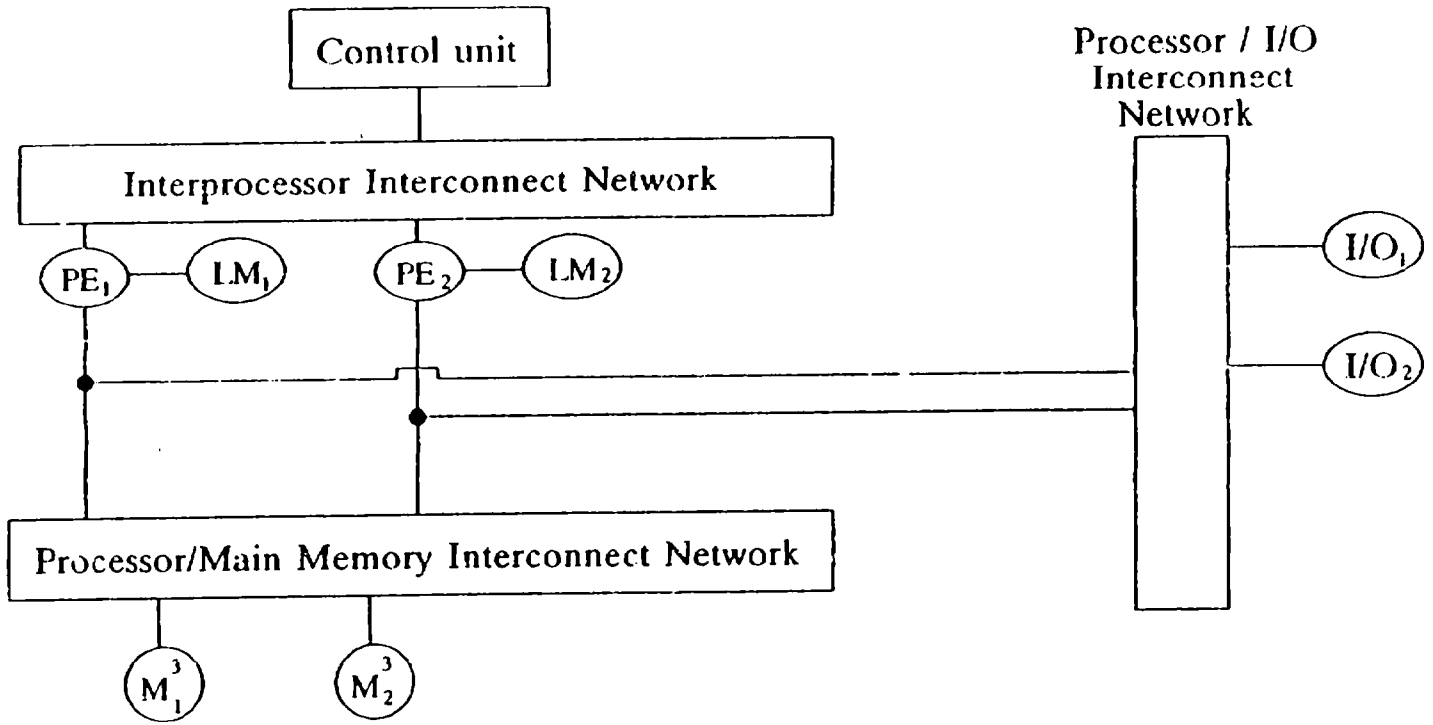
There exist numerous taxonomies for classifying parallel architectures, but this discussion will touch on only those that are deemed useful in the context of this Chapter. For a more complete treatment, the reader is referred to the book by Hwang & Briggs.[24] As a start, one can deal with temporal parallel versus spatial parallel processors. The former most often takes the form of pipelining, which is the sequential execution of instructions or operations such that initial phases of follow-on instructions or operations are initiated before the latter phases of previous instructions or operations are completed. Spatial parallel systems are designed to execute multiple parts of a problem simultaneously. They consist of two or more processing elements, most often of approximately comparable capabilities, such that each element contains at least an arithmetic logic unit and a set of registers. Although the arrangement and connectivity of spatial parallel processors can change, the diagram in

Figure 29 illustrates the general concept behind the architectures. Note that interconnection networks, preferably programmable ones, play a major role in parallel processing.

A classification presented by Seitz,[25] shown in Figure 30, provides an interesting categorization of parallel systems based on the number of processors and the relative degree of processor complexity. Conventional uniprocessor architectures are plotted as a point of reference representing high complexity in a single processor. As one moves up to more than one processor, the trend is toward reduced complexity within each processor, a trend that is driven by total system cost on the one hand and by an escalating overall system complexity on the other hand. Microcomputer arrays are basically a set of computers that send messages to one another via a communication network as illustrated in Figure 31. Such systems are usually loosely coupled (versus tightly coupled); that is, the individual computers do not share main memory and I/O devices, although one computer can always draw upon another's resources through the communication network. The application of such systems in symbolic computing will likely be in solving problems that involve the use of more than one knowledge base. Each processor can work on a given part of the problem in such a way as to minimize the need for interprocessor communications.

The next category, computational arrays, represents systems whose processing elements have been designed for operations on the order of complexity of multiplication and addition. Systolic arrays, for which the processors are connected in regular patterns that match the flow of data in the computation, comprise most of the architectures in this category; however, more general purpose computational arrays will likely emerge as interconnection networks become more flexible. We will return to this point in our discussion of hybrid optical-electronic systems in Section IV.E.

The final categories of parallel machines, logic-enhanced memories and artificial neural systems, can be thought of as "smart memories." Such memories can greatly alleviate the "von Neumann bottleneck," posed by the need in a von Neumann architecture to constantly move information to and



PE = Processing Element
LM = Local Memory
M = Main Memory Module
I/O = Input/Ouput Device

Figure 29. Block Diagram of a General Parallel Computer

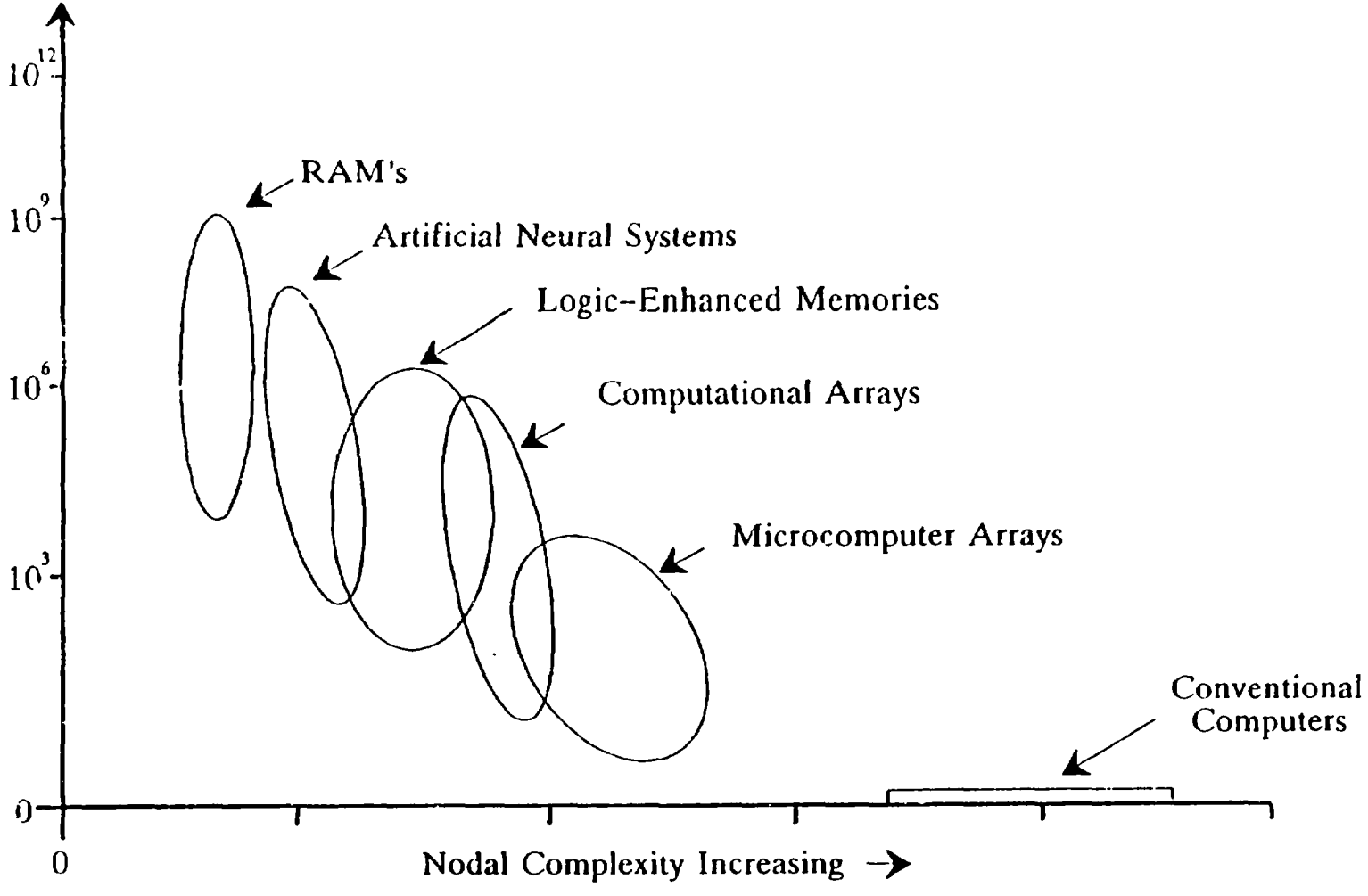
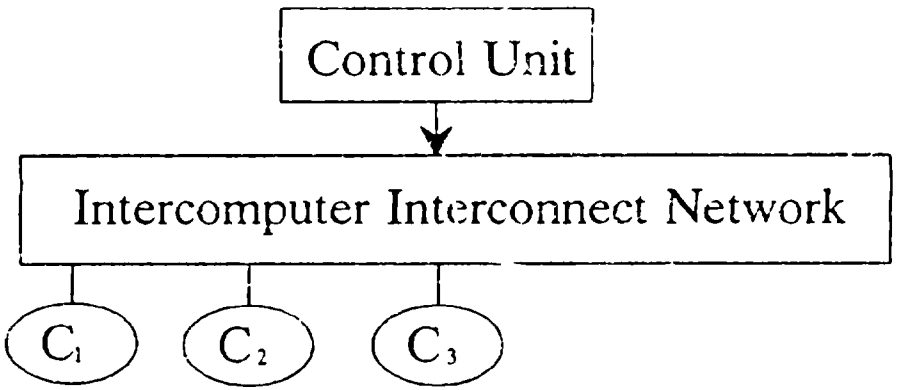


Figure 30. Classification of Parallel Processors by Nodal Company



C = Computer (complete with memory and I/O capability)

Figure 31. Microcomputer Array

from memory, by serving as the memory for a host computer; that is, some of the processing is transferred to the memory to avoid the speed delays incurred by transfers between separated memory and processor units. Each element (or node) of the logic-enhanced memories contains upwards of several thousand bits of storage, a set of registers, and some associated logic capable of operating on the storage contents and of directing communication with the other elements. For the case of artificial neural systems, the complexity of the nodes begins to approach that of a switching element. These systems are referred to as fine-grained parallel processors due to the relatively low complexity of the individual processing elements, and they are always tightly coupled. They must consist of at least several thousand elements to achieve a practical computing power. In fact, some fine-grained architectures with as many as one million elements are currently on the drawing boards.

The final entry shown on Figure 30, that of random access memories (RAMs), is given as a reference on the fine-grained end just as the uniprocessors were shown as a reference for nodal complexity. RAM's, of course, do not function as multiprocessors due to the absence of connectivity.

It is the tightly-coupled fine-grained architectures that are attracting the most interest for symbolic computing because of the emphasis on connectivity over processing power. For example, each node of a semantic network could be mapped to a separate node of such an architecture, and the processing power can be directed toward establishing and identifying the types of the links. Some of the million processor machines mentioned above fall into the category of logic-enhanced memories, and are being considered for handling semantic networks consisting of a few hundred thousand links, and for supporting LISP with a few hundred thousand cons (connection) cells.

One other popular classification of parallel systems deals with Single-Instruction-Multiple-Data (SIMD) versus Multiple-Instruction-Multiple-Data (MIMD). There are two other categories - SISD and MISD - whose definitions should be obvious. SISD represents most of the serial

architectures in existence, and MISD represents a concept which has received very little attention due to its questionable practicality; therefore, only SIMD and MIMD are mentioned in the context of parallel processing. MIMD represents full parallelism and consequently is the most complex of the four categories, requiring individual control units for each of the processors and the ability to efficiently identify and allocate subsets of the problem at hand among the individual processors. The interconnection networks permitting interaction between the units differentiate MIMD systems from systems in which a problem is divided into operations performable on a multitude of SISD machines. Investigations into MIMD architectures have mostly been limited to loosely coupled systems, primarily due to limited knowledge of parallel processing.

The most significant gains in the near future in understanding parallel operations will likely come from implementing SIMD architectures, and therefore they have accounted for most of the current activity in parallel architectures. SIMD does not necessarily mean that all processors are executing the same instruction set but only that they are being presented the same instruction sequence, and each processor can be rendered operable or non-operable by the control unit. For the symbolic operations best suited to large fine-grained machines, SIMD may make more sense than MIMD due to the large overhead incurred by either storing entire instruction sets at each processor or routing instructions through the interprocessor network.

No matter what category given architectures fall into, they all share one overwhelming problem - the need for interconnection networks that can efficiently handle message transfers around the system. The performance of the message-transfer network is a prime factor in determining overall system performance. Figure 29 identified the three basic functional areas in which the networks are important (processor/memory, processor/processor, and processor/IO), and although the performance parameters vary from one to the other depending on the limitations of the components being interconnected, the existing architectures are applicable to any of the areas, and the choices in the past have been made more on the grounds of

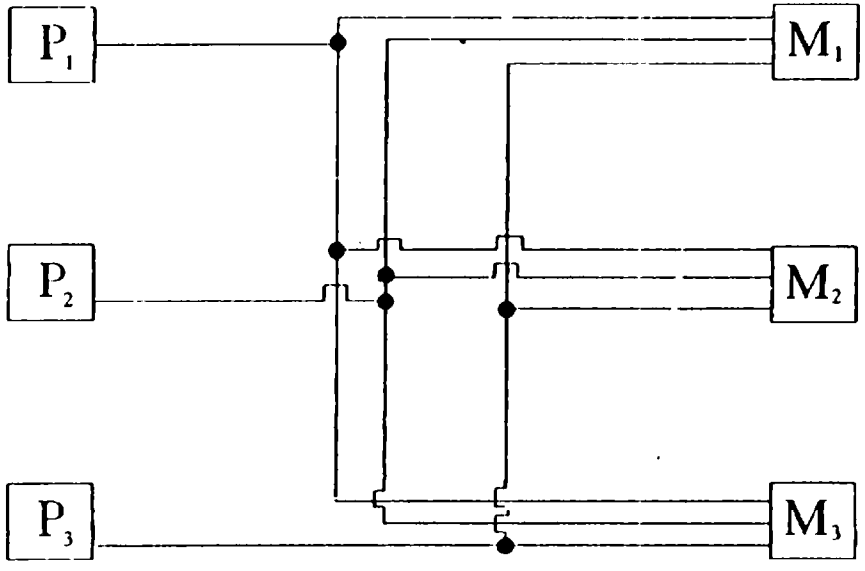
technology limitations and cost rather than functional differences. The following discussion of networks, therefore, will be independent of the functional area.

The thrust in interconnect architectures is toward programmable networks. This is not surprising since the power of the symbolic architectures, especially the fine-grained ones, comes from the interconnects; hence, increasing the flexibility of the nets by making them programmable goes a long way toward enhancing the overall processing power. Furthermore, the system designer is usually willing to trade some speed for the robustness of the slower programmable nets by exchanging hard-wired interconnects for switchable ones which can adapt the network pattern to the data. In symbolic processors, adaptibility is even more important than for numeric processors because the topology of the data structures is usually irregular; therefore, the optimum network topologies cannot be determined prior to system design. In passing, it should be mentioned that programmability is also important from the fault tolerant standpoint since it permits system reconfiguration to bypass faulty processors.

The interconnect networks can be categorized into one of three general classes: multiplexed buses, multiport components, and switching networks. The bus is the simplest, and therefore the most popular, interconnect method because it involves the fewest number of switch elements. However, contention for these switches when many users (processors) are involved limits their utility mostly to loosely-coupled multiprocessor arrays for which the contention for bus resources is not as great as for the more tightly coupled systems.

The contention problem can be lessened by providing components (e.g., memories) with more than one port and by increasing the number of system buses such as shown in Figure 32. This, of course, increases the complexity of the components.

The third class of interconnects uses a network of switches that establish the communication paths around the system. The most general network, a generalized crossbar switch, is capable of establishing independent communication links between all components connected to the



P = Processor
M = Memory

Figure 32. Enhancement of Interconnection via Multi-port Memories

network; that is, there exists no sharing of switching elements between channels and therefore no contention for the switching resources. Although such a network represents the ultimate in interconnect power, its implementation cost in electronic hardware has proven too high for large systems in terms of cost, power requirements, and crosstalk avoidance. A k x m generalized crossbar has k input ports for connection to k data or message sending components and m output ports for connection to m receiving nodes. If each of n nodes of a system were to have one transmitting port and one receiving port (or instead, one bidirectional port), then an n x n crossbar would permit the n nodes to be fully interconnected and free of any contention for network resources.

For the sake of simplicity, a crossbar only of dimension 3 x 3 is illustrated in Figure 33. Note that the cost in terms of hardware for an n x n crossbar would be n^2 switches and $2n^2$ bidirectional communication links. For large n, implementation in electronic integrated circuit technology becomes a formidable problem, especially the design of the $2n^2$ bidirectional links with adequate bandwidth and an acceptable limitation on crosstalk. Also, electronic components have a very limited fan-out capability; for example, the fan-out limitation for an electronic gate is approximately ten other gates. The electronic solution has been to fall back to multi-stage switching networks. An example of one of many possible implementations (a baseline network) is shown in Figure 34a, where each switching element is limited to a fan-in of 2 and a fan-out of 2. The interconnect structure is a three-stage network - all switches in a vertical column would function as one stage of the network. This 8 x 8 interconnect network is composed of twelve 2 x 2 crossbar switches, the functions of which are illustrated in Figure 34b. Only a total of 48 switches (12 x 2 x 2) are needed instead of the 64 needed for an 8 x 8 crossbar. But the multi-stage design leaves the door open for contention. Numerous topologies (e.g., tree, banyan, delta, clos, mesh, to name a few) exist for interconnecting small-dimensional crossbars, and the choice is usually one between cost and an acceptable degree of contention. For example, if the degree of contention associated with a five-stage clos

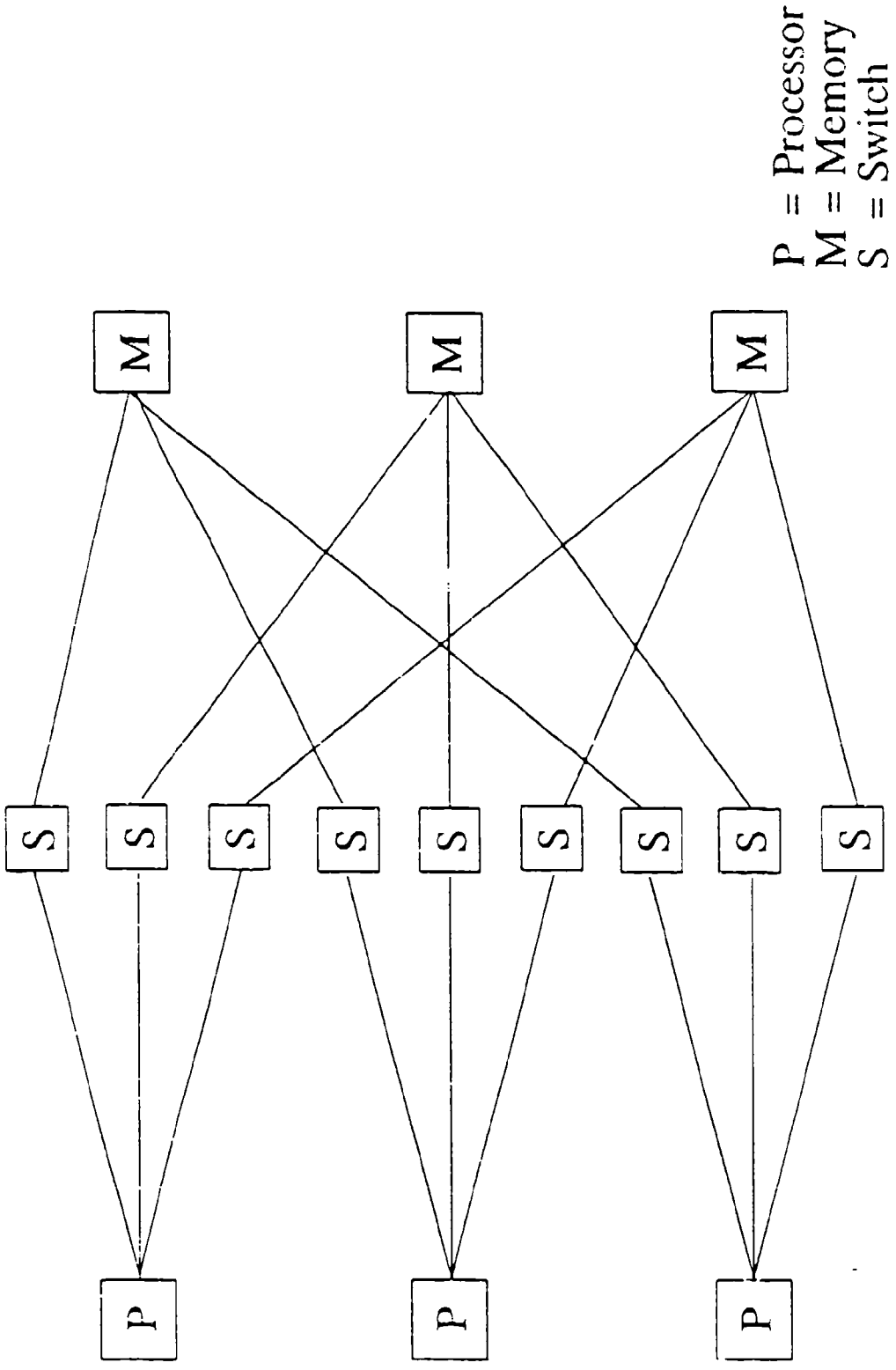
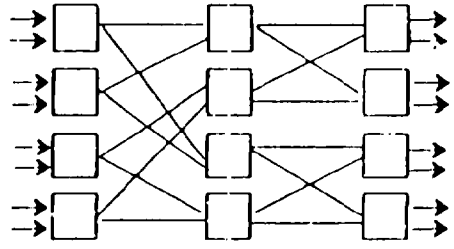
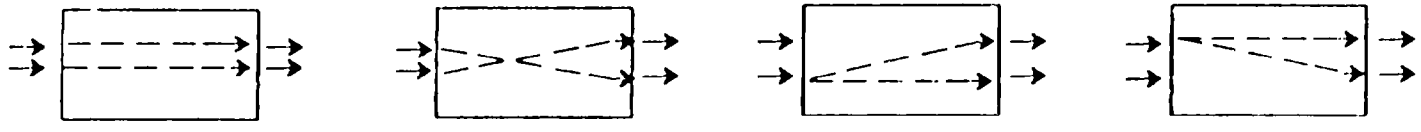


Figure 33. Processor/Memory Interconnect Employing a 3x3 Generalized Crossbar



(a) Baseline Network Topology



(b) Combinations of Switch Settings

Figure 34. A Multi-Stage Switching Network

network is acceptable, a 1000 x 1000 interconnect network would require only 146,300 switches rather than the one million required for the crossbar. A more thorough discussion of network topologies is given by Hwang & Briggs.[24]

Given that a single generalized crossbar is not practical, the network designer must decide what topology best fits the data structure most likely to be encountered. As an example, a prime topology for interconnecting a parallel processor designed for low level image processing would be a grid structure enabling nearest neighbor interconnects since the vast majority of the low level operations involve adjacent image pixels. But as one moves toward the higher operations, regional relationships between the data become more important, requiring more global interconnect topologies.

C. OPTICAL IMPLEMENTATION OF MULTIPROCESSOR ARCHITECTURES

Any approach to optical architectures must give serious consideration to what can be accomplished with existing technologies, namely VLSI electronics. Optical computing will not seriously threaten electronic computing unless it can offer several orders of magnitude improvement in some critical measurement criterion, such as the power-speed-cost product, in a given problem domain. Therefore, a good starting point in addressing the application of optics to symbolic computing is to identify problem areas for electronics.

It is not surprising that the relative weaknesses and strengths of electronics and optics are traceable in one way or another to the fundamental physics of inter-electron and inter-photon interactions. Relatively speaking, the interactions between electrons are strong while that between photons are weak. Hence, electrons are good for the switching operations so fundamental to computing and photons are good for the inter-switch communications, providing links which are free from detrimental coupling effects that lead to crosstalk and capacitive loading. Subscribing to such reasoning, however, is impractical due to the quantum losses which accompany both the electron-to-photon and the photon-to-electron

conversions. There is research and development underway to replace some of the longer interconnect links within computers with optical channels because it is the longer interconnects that create severe power, speed, and space problems for electronics.[26] But such a capability stops far short of using optics to its full advantage in multiprocessor architectures appropriate for symbolic computing.

Consider the electronic-switching/optical-communications position as representing one of the four corners of the square shown in Figure 35 for which the sides of the square represent a continuum of combinations between the extremes of the corners. The upper left corner represents all-electronic systems while the bottom right represents all-optical. Since movement toward the bottom left corner is out of the question, the focus is along the upper and right sides. Upon considering computing systems for which switching is the predominant function, the tradeoff between optics and electronics is seen to fall somewhere along the upper edge; that is, all-electronic switching with some optical links. However, symbolic processing places a strong emphasis on communications as has been pointed out numerous times earlier in the Chapter. The de-emphasis on switching (fine-grained architectures) and the emphasis on communications (tightly-coupled systems) leads one to consider architectures for which the communications is optics and only some of the switching is done with electronics. It is this category of electronic/optical hybrid architectures that we believe will have a significant impact on symbolic computing.

Toward the upper end of this continuum would be those architectures which employ optical switching in the performance of reconfiguring the interconnects but which employ electronic switching for logic operations. As mentioned earlier, optics will prove to be especially valuable in providing the longer (more global) interconnects due to the power, speed, and space penalties associated with the longer electronic interconnects. An example of an architecture with such a designated mixture of optics and electronics is shown in Figure 36. For the sake of simplicity, the illustration shows only two of many possible boards and shows only four chips per board. If this were a fine-grained processor, each chip itself

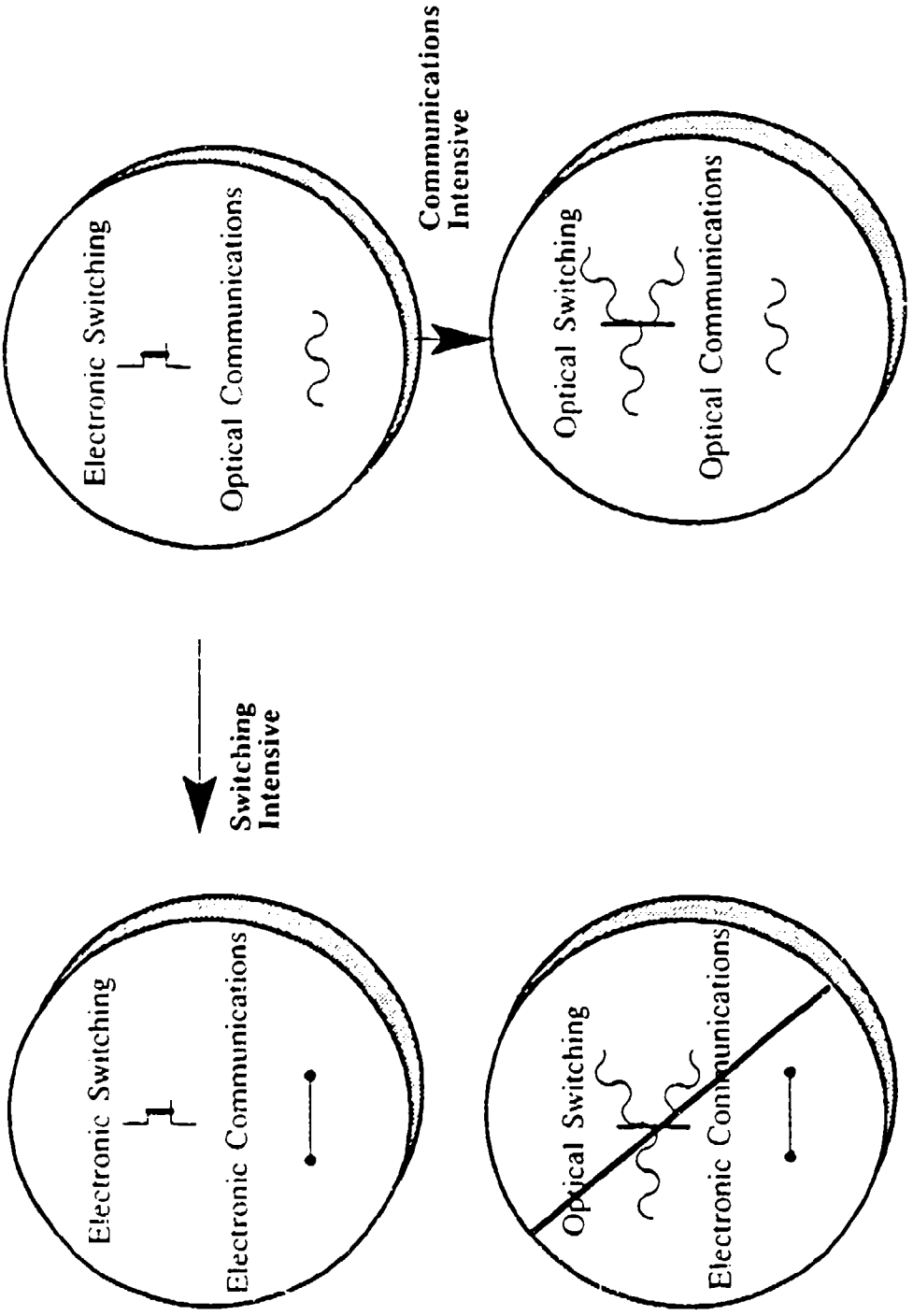


Figure 35. Electronic Versus Optical Computing

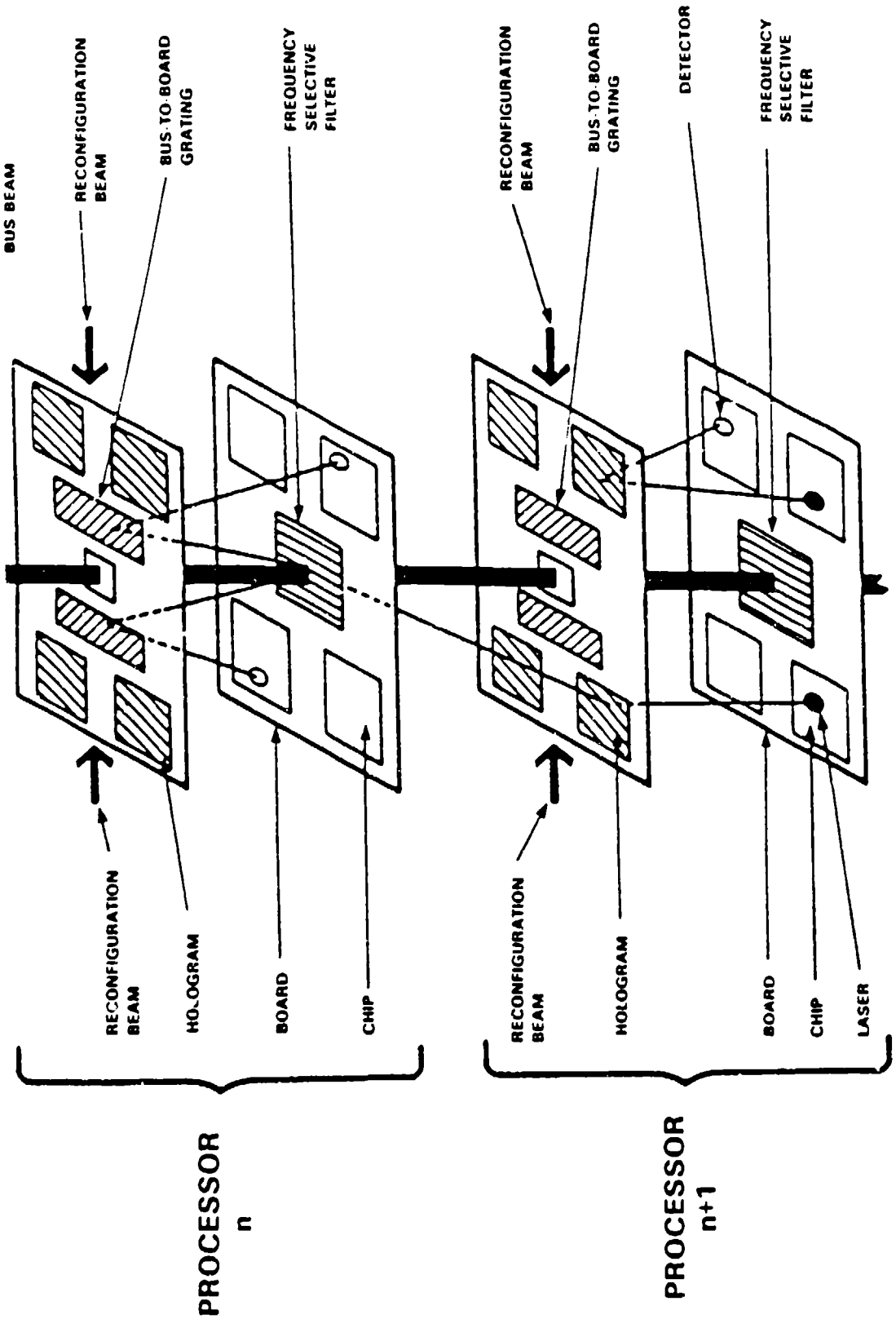


Figure 36. Hybrid/Optical Electronic Multiprocessor Architecture

would contain many processing elements (PEs). For example, one such electronic symbolic computer currently under development, called the Connection Machine,[27] is composed of a large array of printed circuit boards each of which contains 512 PEs equally divided between 32 chips (i.e., 16 PEs per chip).

Each board in Figure 36 contains four optoelectronic chips and one frequency selective filter (hologram). In between each board is a planar array of reconfigurable diffraction gratings that perform the majority of the switching operations involved in the interconnection process. This particular architecture employs wavelength division multiplexing (WDM) to direct optical bit streams to the appropriate board. The beam labeled /1 illustrates this operation. The hologram directly overhead of the transmitting chip directs the beam to the center of the next board where it is superimposed on the main beam which travels to all of the system boards. Upon reaching the intended board, the frequency selective filter diffracts the beam to a bus-to-board hologram which directs the beam to its final destination. The intra-board and intra-chip interconnects would be handled by the plane of holograms above the board as illustrated by beam /2. The logistics of handling a large number of multiplexed beams will not be discussed here other than to say that the optical switching most likely will be achieved through nonlinear wave mixing. For example, four-wave mixing may be used to generate holograms[28] which can be rapidly varied to permit interconnect reconfiguration. The reconfiguration beams shown in Figure 36 would contain the desired information for changing the holographic gratings. Note that some of the switching actions of such an architecture are performed by the multiplexing action, and the various holograms act as passive gratings that selectively direct the various wavelengths.

Such a versatile interconnect scheme based on directing light beams through free space contrasts with one of the most severe problem areas for electronic symbolic computing - that of implementing reconfigurable interconnects. At present, electronics depends on wires for all interconnects, limiting reconfigurability as well as fan-out capabilities. This places a

severe limitation on the switching network architectures, which is complicated by interconnect intensive problems, like those found in AI. The spectrum of optical networks, with their much greater versatility, will not be reviewed here; however, the interested reader is referred to publications of Sawchuk, et. al.[29]

D. ALL OPTICAL ARCHITECTURES

As one moves toward the bottom right corner of the classification of Figure 35, the percentage of optical implementation increases until an all-optical architecture is achieved. An example of a fine-grained, tightly-coupled optical computer is shown in Figure 37. Although no one has built such a computer, it is technically possible to achieve such a system consisting of one million parallel channels. This does not mean that the system would be configured necessarily with one million nodes since such a configuration implies that the planar array of logic elements (designated as the gate array) would have just one logic element per channel. Instead, several logic elements would usually be interconnected via the interconnect media to form a processing element. For example, a square array of n x n logic elements (gates) may comprise an arithmetic logic unit, several registers, and possibly some cache memory. An example of this type of structure is shown in Figure 38, where individual elements in a 2-D SLM have been assigned the necessary functions to comprise a computational processing element. Taking an n of 5 (25 logic elements/processor) would lead to a machine with 40,000 nodes - large enough to be practical as a symbolic computer.

The input to the optical computer could be either through an array of independently addressable laser diodes or a two-dimensional spatial light modulator (2D SLM). The diode array would be capable of much higher modulation speeds, but would involve more complex circuitry, especially if operation requires uniformity over the complete array. If the input already exists as a two-dimensional light pattern such as might be output

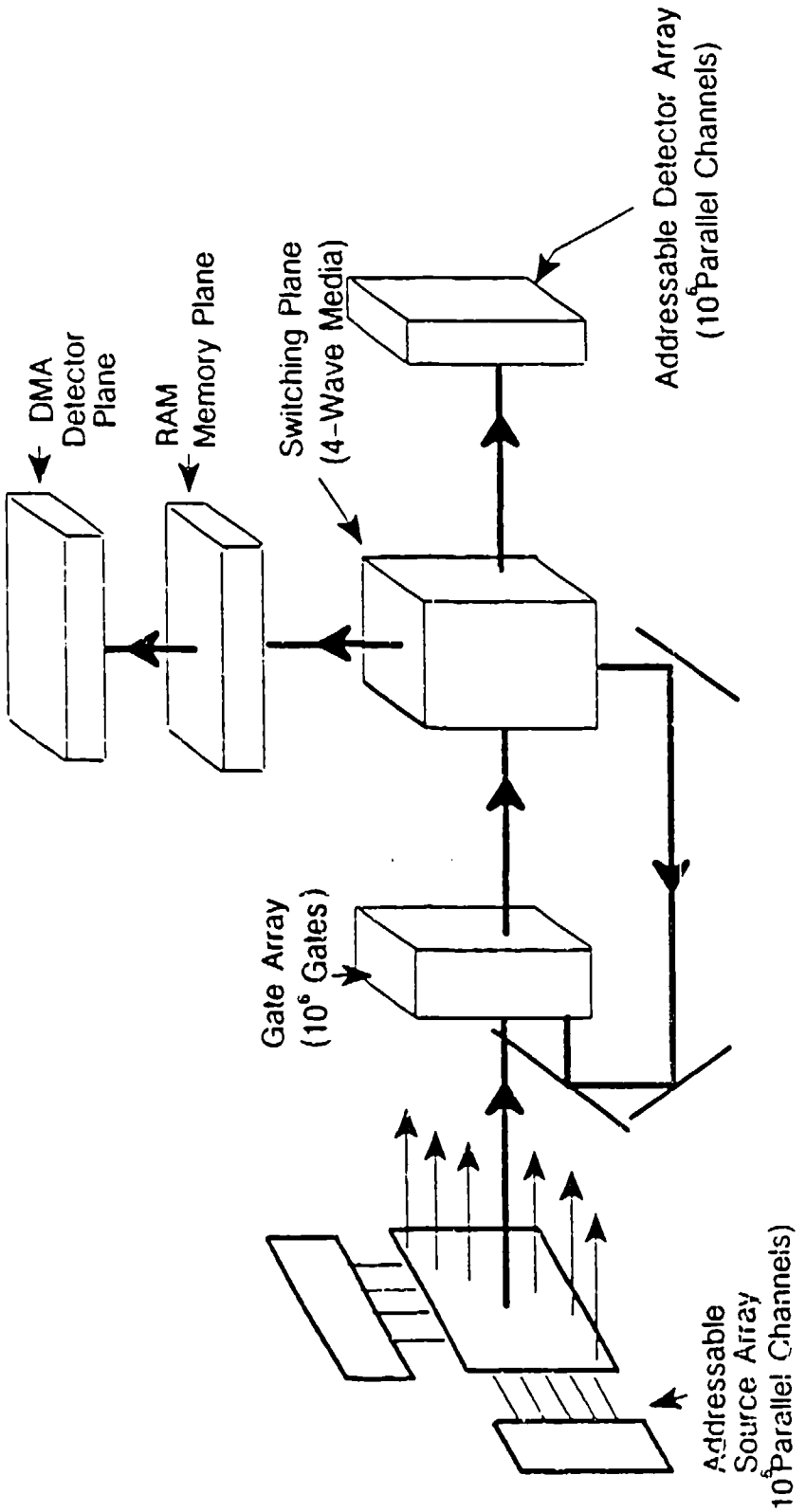


Figure 37. All-Optical Multiprocessor Architecture

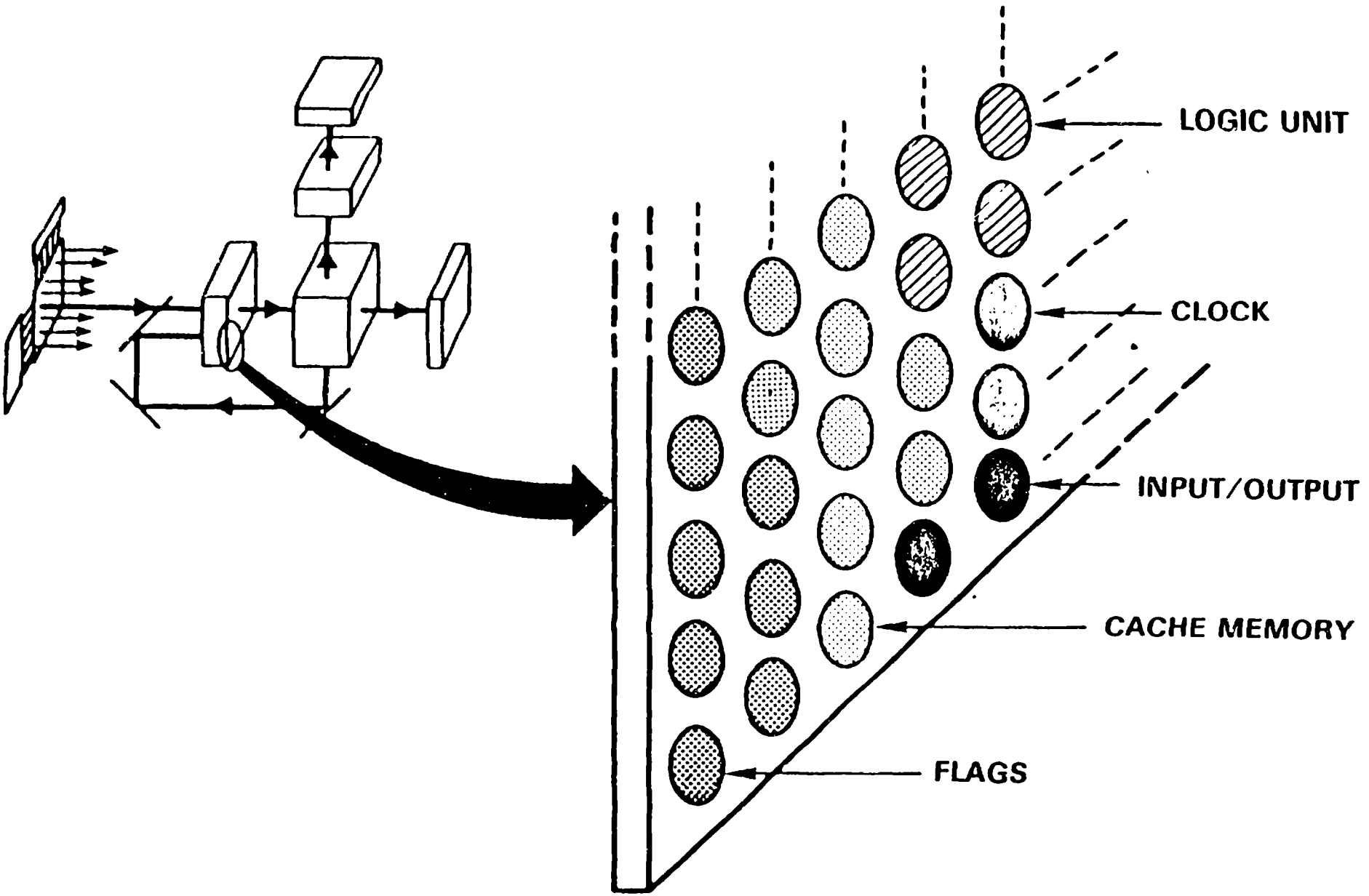


Figure 38. An All-Optical Processing Element

from a vision processor, an input device may not be needed depending on the compatibility of the two processors.

The logic element array could be either a 2D SLM exhibiting a nonlinear response or an array of optical bistable switches. The latter device will ultimately lead to much higher switching speeds, but current realizations of optical bistable switches have required impractical power levels. Improved nonlinear optical materials are currently under development for improved optical bistable devices.

The interconnect element will likely employ wave mixing in a nonlinear optical medium, similar in operation to that described previously for the hybrid architecture. However, due to the much larger number of channels that must be handled, the switching may be done in a multi-stage fashion in which multiple parallel planes of real-time hologram arrays would be exercised as illustrated in Figure 39. Note that, for the sake of simplicity, all three interconnect functions (processor/processor, processor/memory, and processor/IO) are combined into one block, but they could be implemented by three independent devices.

The detector will be a major technological challenge. In the most general case, one would like a one million channel device with each channel operating around 1 MHz (projected speed for 2D SLMs). However, the requirements will be much less for most practical processor designs. If the problem domain were to require, say, 100 iterations or more (e.g., semantic network searches to depths of at least 100), an output would be required only once every 100 microseconds. This reduces the throughput requirements for the detector to 1010 bits per second (bps), a number more in line with projections for GaAs microelectronics. Another example would be where each processor consists of a block of n x n channels as discussed above. Assuming an n equal to 5 and that each processor has just one output channel, the throughput requirement for the detector would be 4.0 x 1010 (bps). Some combination of these two designs should yield a detector requirement that would be well within technical feasibility.

The last major component of this opto-electronic architecture is the memory. The goal of co-locating much of the memory with the logic elements

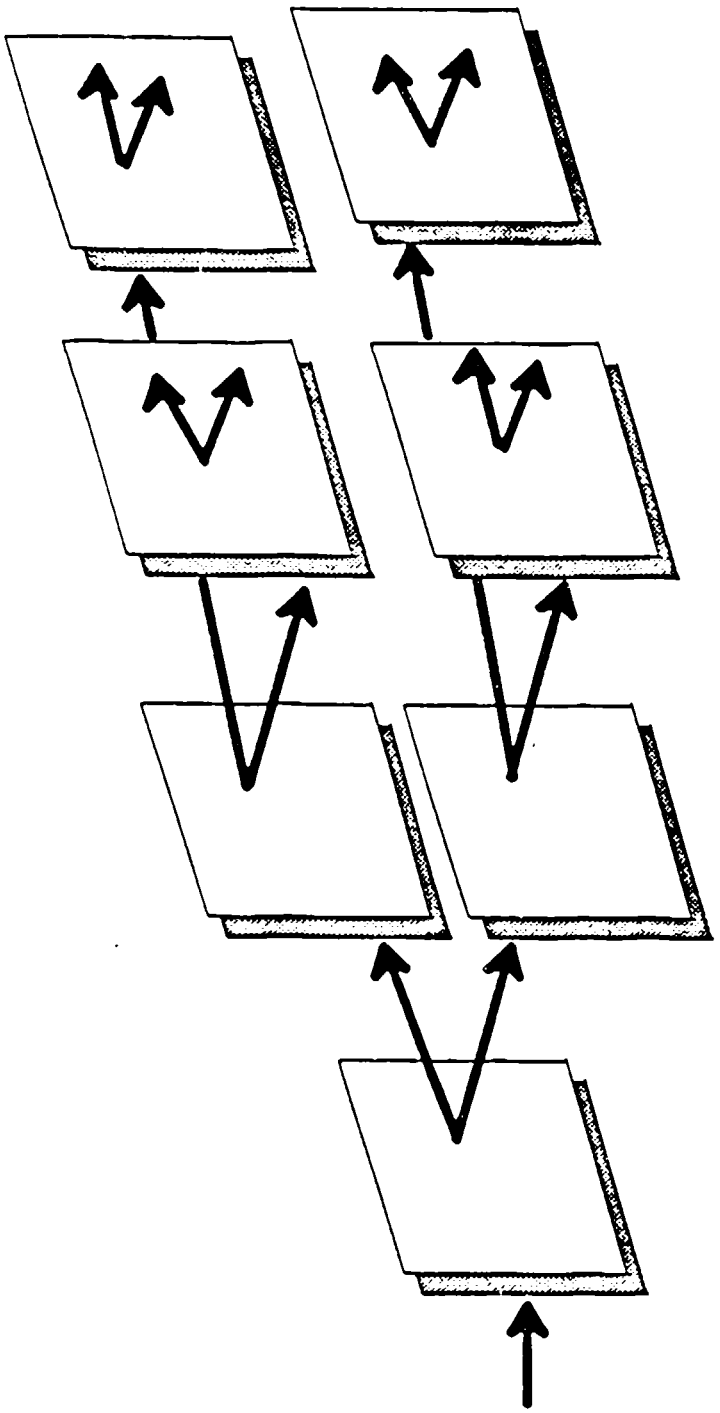


Figure 39. Multi-Stage Optical Interconnect Network

is not necessarily transferable to the optical computer domain because of the greatly reduced communications delays. Thus, Figure 37 shows the main memory as a single block, equally shared by all of the processors. Another important niche for optics is multiport memories. In fact, the use of multiple wavelengths could enable the read-out of any given memory location by a multitude of channels simultaneously, thereby avoiding the need for complex contention-resolving circuitry. For example, holographic gratings could be used to demultiplex the superimposed reflections of a multitude of wavelengths reflected from a given spot on an optical disk, or a holographic memory element could be used that would spatially separate the various read-out wavelengths. A way in which this could be implemented is shown schematically in Figure 40, where multiple beams could be used to address an optical disk simultaneously.

Another appealing attribute of using multiple wavelengths in optical computing is that the switching control is transferred to the information carrying beam itself rather than having to exist as a separate entity, adding greatly to the complexity of the computer control operations. This more closely parallels the operation of message routing systems in which initial bits of the message bit stream contain the address information which is used by each switch that the message encounters as it propagates through the network.

The processing power of the all-optical architecture could be enhanced through the use of pipelining. This could be achieved by replicating the logic element array as shown in Figure 41. Pipelining would be useful for multi-dimensional problems such as vision processing dealing with time-varying three-dimensional imagery (e.g., each plane could handle a different image depth).

By now, the reader should have an appreciation of the types of architectures required for symbolic computation, and of several ways for achieving them optically. It should be clear, however, that even these "all-optical" structures are in some sense hybrid optical-electronic architectures. In particular, electronics would be used for interfaces to the user, to digital controllers, etc, whereas the optics would be used to

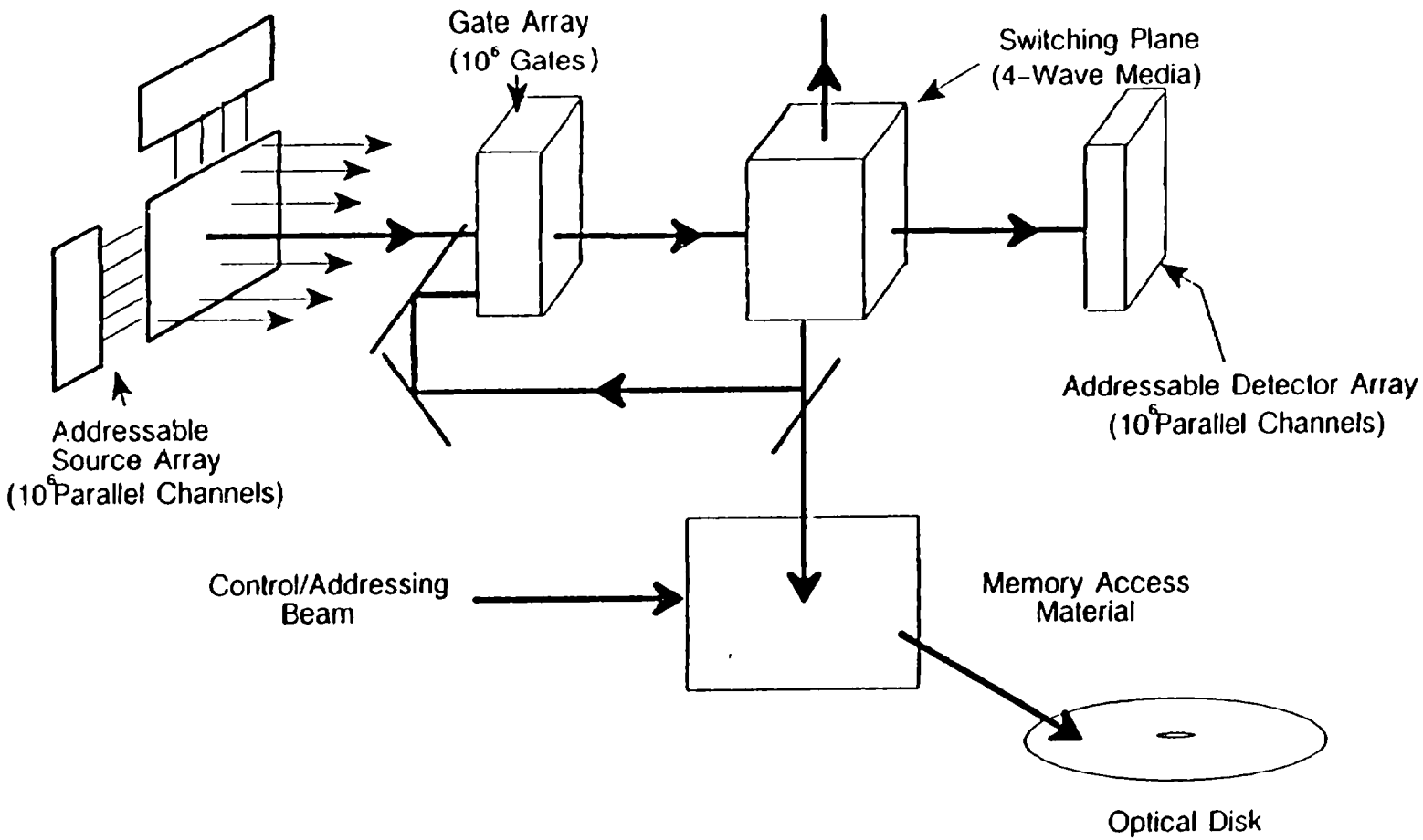


Figure 40. Optical Disk Interface to All-Optical System

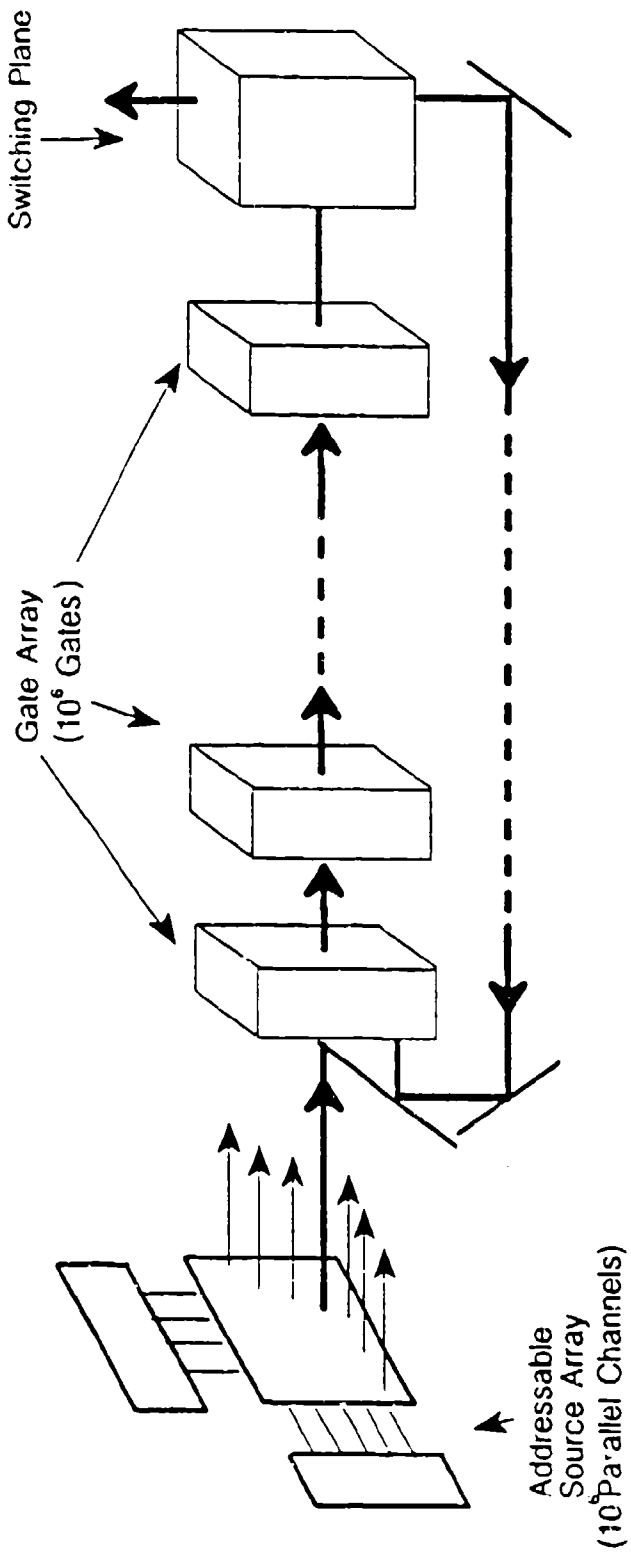


Figure 41. Peipelined Optical Gate Arrays

expedite the symbolic processing. With reference to Figure 36, we would expect that there is a spectrum of levels where the optical-electronic interface could occur. This spectrum of architectural possibilities, ranging from the all-electronic systems, to the hybrid optical-electronic systems, leads to other possible roles for the using optics in symbolic computation. Some of these possibilities will be explored in the following section.

E. HYBRID OPTICAL-ELECTRONIC SYSTEMS

There are several levels on which optics can be effectively combined with electronics. As shown in Figure 42, there is a hierarchy of functions in hybrid systems, ranging from replacing the processor for almost all operations, as in the all-optical systems of the previous section, to the use of optics only as a peripheral device, such as an optical disk for storage. The differences are in the degree of coupling between the electronic processor and the optical system, and in the amount of computation performed by the optics. In the following discussion, we are assuming that the electronic system is the host processor, and that the optical system is connected to it via one of the system busses.

At the lower end, the optical system would entirely replace the electronic system at the processor level, and would therefore perform almost of the computation and would interact strongly with the memory of the electronic system. Using the terminology of Section IV.B, such a hybrid system would be said to be tightly-coupled. At the next level, we have a structure where the optical system computes some of the primitives (multiplication, addition, subtraction, etc.), while the electronics processes others. An example of this could be an optical pattern matcher connected to the data bus of a LISP machine, where the optical system performed all the matching operations, leaving the electronics to compute other primitives. This is also a tightly-coupled system, since the processors share physical memory, and is analogous to the concept of an optical co-processor. The bandwidth of the interconnection network must be

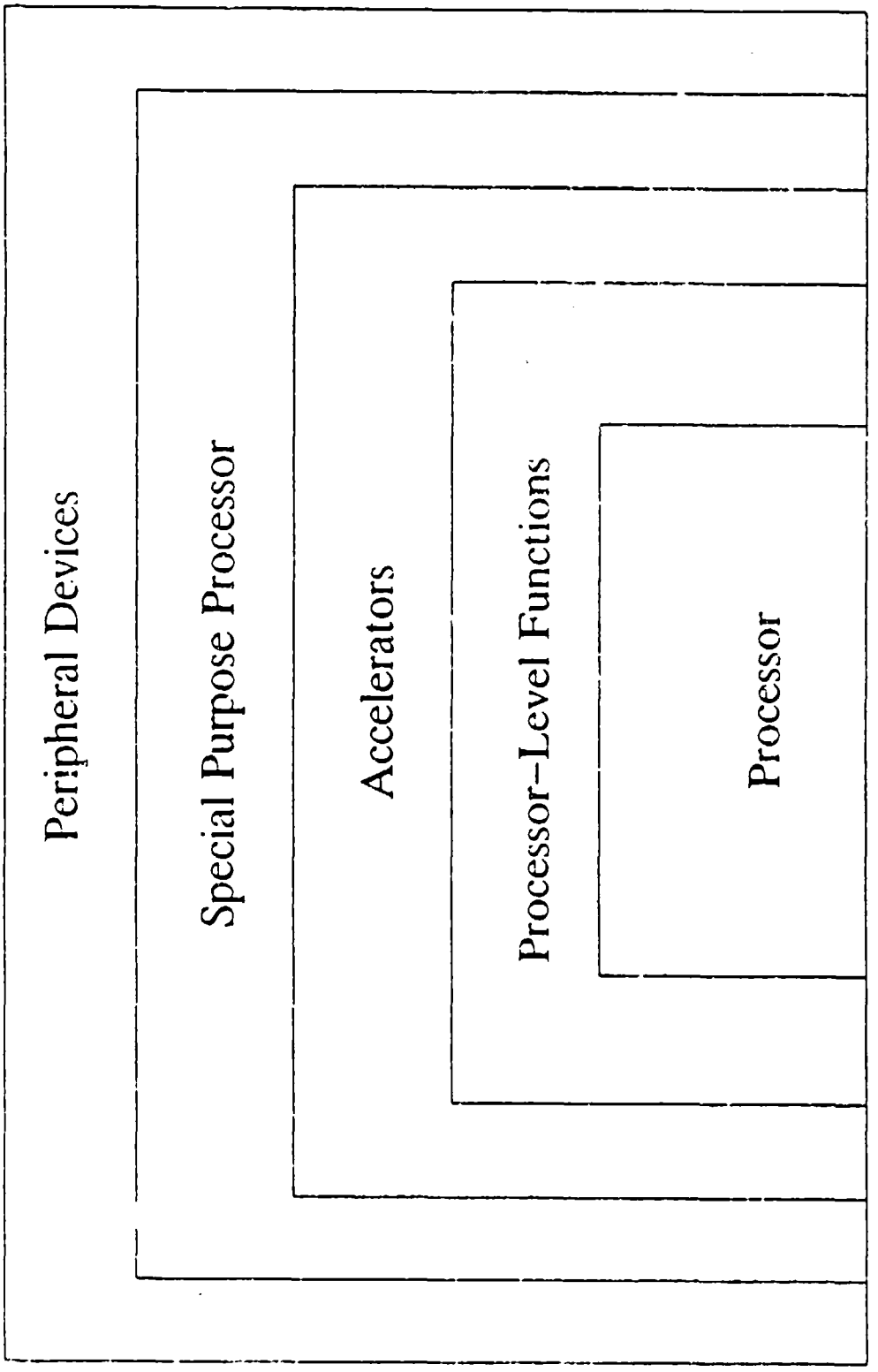


Figure 42. Hierarchy of Functions in Hybrid Systems

very high, and roughly equal to several times the memory access and transfer rate.

Accelerators are processors which dramatically increase the throughput of a particular function, such as an inner product, a correlation, or a rule firing. As we saw in Section III, inner products play a major role in AI processing, such as feature comparison, template matching, and correlation processing at the lower levels, and in knowledge base searches and inferencing at higher levels. Later in this section, we will look at how inner products can be applied to the processing of "if..., then..." types of rules.

Accelerators have had great impact in uniprocessor numeric computation, all but eliminating a number of previously troublesome computational bottlenecks. Surprisingly, they have not yet found widespread application in symbolic computing or in multiprocessor systems, and optics could help hasten that process. As in Figure 44, the optical computer could again be connected to the data bus of the system, but it does not share memory with the electronic system. This is an example of a more loosely-coupled hybrid system, where the host machine and the accelerator are proximally located but not necessarily within the same housing.

Special function processors (SFPs), as the name implies, have traded generality for performance, maximizing the throughput of a specific function. Typically they are very specialized computers with limited programmability, limited memory, and minimal interfacing requirements. As separate computing units, SFPs are connected to the host via a network, an optical fiber, or some other high-bandwidth medium. They are also referred to as computational arrays, and have been used successfully as feature extractors in low-level vision and speech, as display processors in computer graphics, and as array processors for computing FFTs. The most popular SFP is the systolic array, for which there is a rich base of experience, and a number of optical implementations as well.

For the remainder of this section, we would like to focus on two examples from the discussion above, namely the use of optical computing as an accelerator and as a special purpose processor. For the accelerator

case, we will use the example of inner product processing for "if..., then..." type rules. As a special purpose processor, we will look at the potential role of systolic array implementations of semantic net processing.

A central operation in symbolic computing is the inner product, which is equivalent to a multiplication of component elements in a vector (vector multiplication), in a matrix (matrix-matrix multiplication), or in a correlation function. In earlier sections, we identified the commonality of inner products in a large number of algorithms in the numeric computing domain. In one typical symbolic computing representation, knowledge relations are expressed in terms of logical pattern matching, such as determining the agreement of an antecedent condition (left-hand side) of an "if A , then B " relation (see Section III.E). Here A takes the form of a vector subspace of a N-dimensional vector space:

A = data/objects that belong to class A

which is spanned by some M vectors, where M s N:

ai(k) n A , k = 1,..., M

Membership in this subspace can be verified by means of a simple functional operation. For example, the null functional of a subspace is uniquely represented in terms of the vector aA that is orthogonal to the subspace a(k). Then, the inner product:

aA , ai(k) = 0 for all ai(k) n A

Thus, a calculation, or, in this case, a rule firing, between knowledge elements in this representation and some appropriate functional may be expressable in terms of inner product functions. Furthermore, this construct could be used in either forward-chained or goal-directed reasoning, since in each case, the antecedent of an "if..., then..."

production rule must be satisfied during an inference. This is also true for frame-based representations, since, in that case, each knowledge element is a 2D array of information processed in its entirety, and the ai s may take the form of matrix components. For each rule firing, the matching operation can be computed on the optical system very efficiently, alleviating a serious computational bottleneck in several AI systems.

Inner products are not the only operations for which optics may have a role. Systolic arrays, of which there have been several optical implementations, 7-9 have been shown to have definite mappings onto signal-flow graph networks. This implies that problems treatable or based on graph-theoretical techniques may have direct mappings onto well-defined systolic array topologies. In symbolic computing, graph theory analysis has been applied to developing relationships between definable objects and their attributes; this research has led to the semantic net representation.

For purposes of illustration, the semantic net can be viewed as a collection of nodes representing symbols, which are connected by links representing relations. The most fundamental relationship between symbols is the "IsA" link, and other types of relations could be "AtLocation," "MemberSet," and "Partof." Such relations are domain specific, and depend upon the taxonomy of the problem under investigation. In this representation, a basic question common to symbolic computation systems would be "Is A a B ?." If we assume that all connections between a general class of nodes S are constituted by "IsA" links, then this query is reducible to the problem of:

"For A a member of S ,"
"Is B also a member of S ?" and
"Is there a connectivity between A and B ?"

On a conventional architecture, this query would involve an extensive sequential search over all memory paths emanating from A . However, on a systolic system, the conclusion involves only the number of steps between

A and B , or to the edge of the array (for B not an element of S); this is because the search would proceed along all branches simultaneously.

To map this question onto the systolic system, we can define each node to reside on a processor or a pixel, and the links to adjacent nodes are the existing topological connectivity of the array. Here, the array is a 3-D construct with the third dimension being time. Node A can be tagged as the element of interest, and the resulting search towards B can occur as internodal bit stream propagation in each time step. This is shown schematically in Figure 43. While this still requires some test for the match condition, the set of semantic net problems should have some direct mapping onto systolic array systems.

In summary, optics offers several unique capabilities for the generic architectures discussed above, architectures which hold great promise for symbolic computing. The most important of these capabilities are: high speed global interconnects, interconnect reconfigurability, high fan-out elements, and multiport components for parallel processing.

F. ACKNOWLEDGEMENTS

The authors would like to express their gratitude to a number of personnel and co-workers who took time away from their own activities to work with us to improve the quality of our manuscript. While their are too many to cite individually by name, some deserve special credit. At the top of the list are Drs. Ravindra A. Athale, Teresa A. Blaxton, Roger A. Geesey and Lawrence H. Reeker, whose thoughtfulness and timely insights were extremely valuable. Many of their thoughts, explanations, and interpretations have found their way into our manuscript. John Perry, for his aid in the image understanding area and for agreeing to let us use some of his computer-generated images. Thomas S. Stark, for developing the glossary, and helping us bridge the "terminology-gap" that exists between optics and symbolic computing. And finally, Dr. Charles T. Butler, for assisting us in a number of ways, providing the authors with the time to develop our ideas more thoroughly.

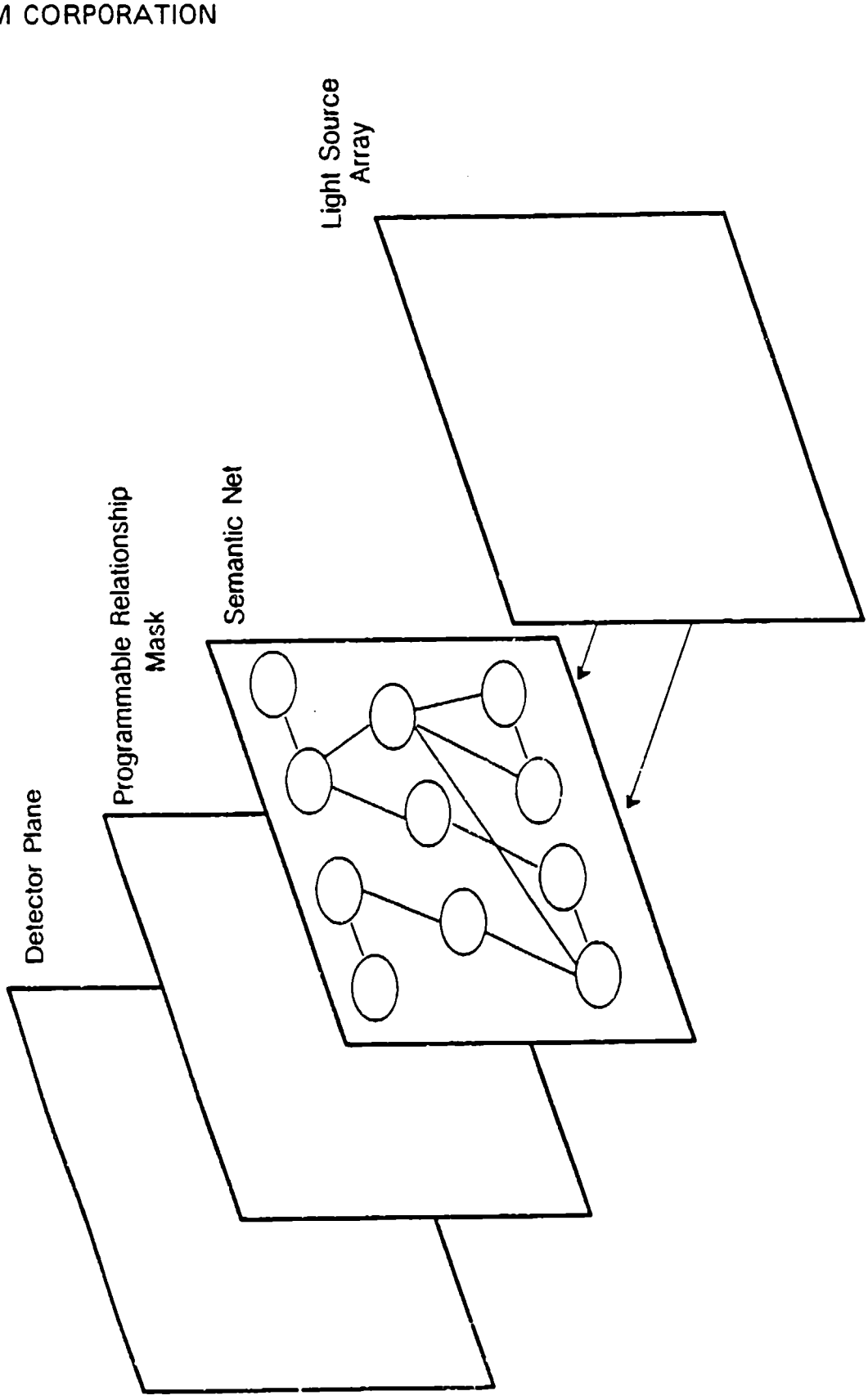


Figure 43. Parallel Processing of a Semantic Net

CHAPTER V
GLOSSARY

AI LANGUAGES AND TOOLS — A principle of Artificial Intelligence according to N. J. Nilssons Onion Model.

AGENDA — A part of an expert system that contains a prioritized list of knowledge based rules awaiting execution.

ANALOGICAL KNOWLEDGE — Knowledge that is represented by analogy in a computer. Examples of this are sound patterns representing words in a natural language processing system, or the representation of an image by a 2D array of numbers corresponding to steps on a gray scale.

ARCHITECTURE — The organization of the individual elements of a computer.

ASSERTION — A positive statement or declaration.

ATOM — A symbol (either constant or variable) used to identify an object in a LISP program.

ATTRIBUTE — Part of the description of an object contained in a frame. Attributes normally tell characteristics such as color, size, and value. The same as a slot.

GLOSSARY (CONTINUED)

BACKWARD CHAINING — A recursive procedure for problem solving in knowledge based systems. A progression by goal-driven inference is attempted between an assumed goal state and one or more initial states. If the progression is not possible, the goal state is altered. If the progression is possible, the initial state or states become goal states and the procedure begins again.

BACKTRACKING — A search procedure in which guesses are made about the direction to be taken through the solution space. When the guesses lead to an unacceptable result, the procedure backtracks to the point at which the incorrect guesses were made and begins the search procedure again in alternative directions.

BELIEF — A hypothesis about the outcome of some unobservable or uncertain situation.

BLACKBOARD — A part of many artificial intelligence systems in which intermediate or partial results of problem-solving are recorded.

GLOSSARY (CONTINUED)

BREADTH FIRST SEARCH	A search strategy used in knowledge based systems where the possible solutions to the problem are represented by a tree with nodes and branches. In breadth first search, all branches of the tree are examined at one node or level before moving to the next level. In this way, searching the breadth of the solution space is emphasized. See Depth First Search.
BOTTOM UP	A method of problem-solving which progresses from an initial condition to some desired condition. See Data directed inference and forward chaining.
CERTAINTY	A measure of the confidence placed by a user on the validity of a proposition, hypothesis, or inferential rule.
COMMON SENSE REASONING AND LOGIC	A principle of AI from Nilsson's Onion model.
COMPUTATIONAL LOGIC	Logical reasoning done on a symbolic computer. This is the basis for non-numeric computations done in AI systems.

GLOSSARY (CONTINUED)

CONTINUOUS SPEECH	Normal spoken language. The majority of speech recognition systems accept either isolated or continuous speech.
CONTROL	The determination of the overall order or organization of problem solving procedures or activities.
DATA-DIRECTED INFERENCE	The type of inferences employed in forward chaining. By applying inference rules to supplied data or conditions a logical result is derived.
DECLARATIVE KNOWLEDGE	Knowledge consisting of facts or assertions.
DEPENDENCY	The relation between logical conclusions and the premises and inference procedures from which they were derived.
DEPTH-FIRST SEARCH	A search strategy in knowledge based systems in which possible solutions to the problem are represented by a tree with nodes and branches. In depth first search, a branch of related solutions is considered at all nodes before moving to the next branch. In this way, the depth of an assumed class of solutions can be examined before moving to the next branch. See Breadth first search.

GLOSSARY (CONTINUED)

DOCUMENT GENERATION	One of many proposed applications of natural language processing systems. After information is stored in a computer, the computer generates a document containing the information.
DOCUMENT PREPARATION	Another proposed application of natural language processing systems. NLP systems act as experienced editors, checking for errors in spelling and grammar and suggesting ways to rephrase text.
DOCUMENT UNDERSTANDING	A proposed application of NLP systems in which a document is read and its contents are assimilated by the system.
EXPECTATION-DRIVEN REASONING	A control procedure that uses expectations to formulate hypotheses about unobserved situations. See backward chaining and goal-directed inference.
EXPERT FRAMEWORK SYSTEMS	The knowledge representation and reasoning mechanisms of an expert system without the domain specific knowledge base.
EXPERT SYSTEM	A computer system that achieves high levels of performance in areas that for human beings would require years of special education and training.

GLOSSARY (CONTINUED)

EXPERT SYSTEM DEVELOPMENT ENVIRONMENT	The knowledge representation and reasoning mechanisms of an expert system without the domain specific knowledge base.
EXPERTISE	The capabilities that enable high performance in a particular area. In the context of AI systems, this includes tools that enable intelligent operation such as meta-knowledge, heuristic search procedures, and inference rules.
EXPLANATION SUBSYSTEM	A part of an expert system that rationalizes or explains its conclusion by providing a summary of the inference rules and data that it used to arrive at the conclusion.
FACT	A piece of declarative knowledge.
FORWARD-CHAINING	A procedure for problem solving in knowledge based systems. The procedure progresses from the data given by the user to a solution that adequately supports it. See data driven inference.
FRAME	Data structures that represent objects by a list of properties and relations to other objects.

GLOSSARY (CONTINUED)

FUZZY LOGIC	An approach to inexact reasoning consisting of a proposition and a fuzzy set. The fuzzy set contains ranges of possible values the proposition can have and numerical values corresponding to the probability of occurrence in each range.
GOAL-DIRECTED INFERENCE	The type of inferences employed in backward chaining. Emphasis is placed on examining the states and inference rules which produce a desired goal. See backward chaining and expectation-driven reasoning.
GRAY SCALE	Analog or digital numbers corresponding to shades of gray. The maximum and minimum numbers represent white and black.
HEURISTIC	Of or relating to problem solving procedures that utilize self educating techniques to improve their performance.
HEURISTIC SEARCH	A principle of AI from Nilsson's Onion model.

GLOSSARY (CONTINUED)

HIERARCHICAL PLANNING	An approach to planning in which computer time and memory are saved by considering only high level details of the plan. Vague and lower level details are then formulated into subplans.
HUMAN ENGINEERING	The process of engineering man-machine interfaces to achieve maximum utilization of that machine.
INFERENCE	The process of passing from a proposition whose truth is established to another whose truth is believed to follow from that of the former.
INFERENCE ENGINE	A part of an expert system containing the procedures it will use to solve problems.
INFERENCE RULES	Methods or strategies employed by AI systems for solving problems by logical inference.
ISOLATED SPEECH	Speech that has pauses between the words. The majority of speech recognition systems accept either isolated or continuous speech.

GLOSSARY (CONTINUED)

KNOWLEDGE	Stored information for use ir the solution of AI problems. In the case of expert systems, knowledge is thought of as facts, beliefs, or heuristic rules. In the case of speech and image recognition systems, knowledge is thought of as; stored pattern, image, or word data, beliefs, and heuristic rules.
KNOWLEDGE ACQUISITION	The extraction and formulation of knowledge for use in AI systems. The AI computer equivalent of learning.
KNOWLEDGE BASE	The repository of knowledge in an AI computer system.
KNOWLEDGE ENGINEERING	A discipline associated with the building of expert systems.
KNOWLEDGE PROGRAMMING TOOLS	Software that allows an expert who knows little about knowledge engineering to program knowledge into an AI system.
KNOWLEDGE REPRESENTATION	A principle of AI from Nilsson's Onion model.

GLOSSARY (CONTINUED)

LEARNING	The process of improving the performance of an AI system by using past experience to alter its stored knowledge or problem solving strategies.
LEXICON	A list of morphemes contained in a NLP system's knowledge base.
LOGIC ORIENTED PROGRAMMING LANGUAGES	Programming languages whose purposes are to program and solve logic problems.
LISP	The most widely used AI programming language. LISP was developed by John McCarthy at MIT in 1958. LISP is an object oriented programming language.
LISP MACHINES	Computing architectures dedicated to executing LISP programs.
LIST	A series of elements (either atoms or other lists) enclosed in parentheses in a LISP program.
MACHINE TRANSLATION	A proposed application of natural language processing systems for the translation of a document from one language to another.

GLOSSARY (CONTINUED)

META	A prefix used with AI subjects to denote the existence of knowledge about the base word or subject, as in meta-knowledge, which is knowledge about the system's knowledge base.
MORPHEMES	A basic linguistic unit having meaning.
MORPHOLOGICAL ANALYSIS	A technique used in natural language processing. The meaning and use of a word is determined by separating the word into parts and determining the meaning of each part.
NOISY DATA	Data having characteristics that introduce uncertainty into the reasoning processes of AI systems.
OBJECT-ORIENTED PROGRAMMING LANGUAGES	Programming languages whose purposes are to portray the characteristics and relationships between objects.
PARSING	The act of breaking a sentence or phrase into its component parts in order to identify the form, function, and syntactical relationship between each part.

GLOSSARY (CONTINUED)

PRAGMATICS	Knowledge about human discourse and conversations, concerning the overall context in which sentences or phrases are written or spoken and how the various phrases are related to each other.
PREDICATE CALCULUS	A formal language of symbol structures for representing facts.
PREDICATE LOGIC	Logical operations that, through the manipulation of propositions, allow one to make assertions.
PROCEDURAL KNOWLEDGE	Knowledge about procedures or actions, typically specified as If..., then.... types of production rules.
PRODUCTION RULES	Two-part statements that specify a certain action to be taken when an antecedent condition is satisfied, usually in the form of If..., then... types of rules. Procedural knowledge is contained in production rules.
PROLOG	PROgramming in LOGic, was developed by A. Colmerauer and P Roussel at the University of Marseille in 1973. Prolog is a logic oriented programming language.

GLOSSARY (CONTINUED)

PROPERTY LIST	A construct in a LISP program that associates a property and a corresponding value with each atom. Property lists describe the "state of the world" in an AI program and therefore are updated frequently.
PROPOSITIONAL LOGIC	Logic which, through the use of propositions, determines the truth or falsehood of other propositions.
PRUNING	The act of eliminating a solution or group of solutions from a problems solution tree.
RULE	"If...Then" statements that support deductive reasoning.
RULE SET	A collection of rules that constitutes a module of heuristic knowledge.
SCHEDULING	Determining the order of execution of processes in AI programs.
SCRIPTS	Data structures that represent sequences of events. Scripts represent procedural knowledge and are similar in principle to frames.

GLOSSARY (CONTINUED)

SEMANTICS	The non-literal interpretation of word meanings. Knowledge in this area is used extensively in speech and natural language interpretation.
SEMANTIC NETWORK	A scheme for representing relationships between objects in an AI system's knowledge base in terms of class equivalence and inheritances.
SLOT	A single description of an object in a frame. Slots can contain information such as name, color, definition, or value.
SOLUTION SPACE	A conceptual way of thinking of the possible solutions to a problem, which has a direct bearing on the number of branches that an AI system can search in the course of solving a problem.
SOLUTION TREE	The representation of the possible solutions to an AI problem by a tree with nodes representing solution types, and branches representing their relationships.
SPEAKER DEPENDENT SPEECH RECOGNITION SYSTEMS	Speech recognition systems that can only accept input from a particular speaker.

GLOSSARY (CONTINUED)

SPEAKER INDEPENDENT SPEECH RECOGNITION SYSTEMS	Speech recognition systems that can understand input from any speaker without being trained to each individual voice.
SPEECH RECOGNITION	The recognition of spoken language by a computer system.
SYNTAX	The use of words to form phrases, clauses, and sentences.
TEMPLATE MATCHING	One way in which images and spoken words are identified. Arrays representing frequency (sound for words, light for images) or intensity versus time are matched against those of known images and words for similarity.
TOP DOWN	An approach to problem solving which moves from some current condition to an initial condition. See **BACKWARD CHAINING** and **GOAL DIRECTED INFERENCE.**
TRUTH MAINTENANCE	The act of maintaining truth or consistency in the elements of a knowledge base.
VISION	The understanding and interpretation of scenes or images by a computer system.
WELL FORMED FORMULA (WFF)	A syntactically valid statement in predicate calculus.

CHAPTER VI
REFERENCES

1. C. L. Forgy, et al, "Opportunties for Parallelism in AI Tasks," Carnegie-Mellon Reports, 1985.

2. H. Winston, Artifical Intelligence, Addison-Wesley, Reading, MA 1984.

3. E. Charniak, D. McDermott, Introduction to Artificial Intelligence, Addison-Wesley, Reading, MA, 1985.

4. Fennell and Lesser, "Parallelism in Artificial Intelligence: A case study of Hearsay II," IEEE Trans On Computers, Volume C-26, No. 2 (February 1977) 98-111.

5. D. Marr, Vision, Freeman, San Francisco, 1982.

6. A. Barr, E. Feigenbaum, et al, eds. The Handbook of Artificial Intelligence, 3 Volumes, Kaufman, Los Altos, CA 1981.

7. R. Lindsay, B. Buchanan, E. Feignebaum, J. Lederberg, Applications of Artificial Intelligence for Organic Chemistry: The Dendral Project, McGraw-Hill, New York, 1980.

8. J. McDermott, "R1: The Formative Years," AI Magazine, 2, No. 1 (Spring 1981) 21.

9. E. H. Shortliffe, Computer-Based Medical Consultations: MYCIN, American Elsevier, New York, 1976.

10. F. Hayes-Roth, D. Lenat, D. Waterman, Building Expert Systems, Addison-Wesley, Boston, MA, 1983.

11. H. L. Andrews, "Speech Processing," IEEE Computer, Volume 17, No. 10 (October 1984) 315.

12. R. Balzer, L. Erman, et al, "HEARSAY-III: A Domain Independent Framework for Expert Systems," Proc. AAAI-1 1980

13. D. H. Ballard and C. M. Brown, Computer Vision, Prentice Hall, Englewood Cliffs, NJ, 1982.

14. R. D. Duda and P. E. Hart, Pattern Classification and Scene Analysis, Wiley, New York, 1973.

15. F. Itakura, Minimum Prediction Residual Principle Applied to Speech Recognition, IEEE Trans. Acoustics, Speech, and Signal Processing, Volume ASSP-23, (February 1975) 67.

REFERENCES (CONTINUED)

16. See, for example, references in Artificial Intelligence. 17, No. 1-3, (Special Issue on Vision) 1981.

17. K. Fu and A. Rosenfeld, "Pattern Recognition and Computer Vision," IEEE Computer, Volume 17, No. 10 (October 1984) 274.

18. E. Rich, Artificial Intelligence, McGraw-Hill, New York, 1983.

19. R. Davis, D. Lenat, Knowledge-Based Systems in Artificial Intelligence, McGraw-Hill, New York, 1982.

20. See, for example, T. A. Blaxton and B. G. Kushner, "An Organizational Framework for Building Adaptive Artificial Intelligence Systems," in preparation.

21. B. G. Buchanan, D. H. Smith, et al, "Applications of Artificial Intelligence for Chemical Inference XXII: Automatic Rule Formation in Mass Spectrometry by Means of the Meta-DENDRAL Program," Journal of the American Chemical Society, Volume 98, (1976) 6168.

22. D. Lenat, "EURISKO: A Program that learns New Heuristics and Domain Concepts," Artificial Intelligence, 21, No. 1, (January 1983) 31.

23. J. R. Anderson, The Architecture of Cognition, Harvard University Press, Cambridge, MA, 1983.

24. K. Hwang and F. A. Briggs, Computer Architecture and Parallel Processing, McGraw-Hill (1984).

25. C. L. Seitz, "Concurrent VLSI Architectures," IEEE Transactions on Computers, Volume C-33, No. 12 (1984).

26. J. A. Neff, "Optical Interconnections Between Integrated Circuit Chips." Proceedings of the International Electronics Packaging Conference, Orlando, FL (1985).

27. W. D. Hillis, The Connection Machine, The MIT Press (1985).

28. D. M. Pepper, "Nonlinear Optical Phase Conjugation," Optical Engineering, Volume 21, No 156 (1982).

29. A. A. Sawchuk, B. K. Jenkins, C. S. Raghavendra, & A. Varma, "Optical Interconnection Networks," Proceedings of the IEEE International Conference on Parallel Processing, St. Charles, IL (1985).

REFERENCES (CONTINUED)

30. H. J. Caulfield, W. T. Rhodes, M. J. Foster, and S. Horvitz, "Optical Implementation of Systolic Array Processing," Opt. Commun., Volume 40, No. 12 (December 1981) 1986.

31. P. S. Gulfoyle, "Systolic Acousto-optic Binary Convolver," Opt. Eng., Volume 23, No. 1 (January/February 1984) 20.

32. S. Cartwright, S. C. Gustafson, "Convolver-based Optical Systolic Processing Architectures," Opt. Eng. (January/February 1985) 59.

33. John Perry, "Image Understanding by Computers: Algorithms and Implementations" BDM Technical Report BDM/W-86-0131-TR, February.

34. J. L. Baer, "Computer Architecture," IEEE Computer, Volume 17, No. 10 (October 1984) 77.

OPTICAL SYMBOLIC COMPUTING

Brian G. Kushner* - John A. Neff**

* The BDM Corporation, 7915 Jones Branch Drive, McLean, VA 22102-3396
** Defense Advanced Research Projects Agency, 1400 Wilson Boulevard Arlington, VA 22209-2308

ABSTRACT

The need to drastically increase the processing rate of artificial intelligence (AI) systems, coupled with the limitations of current uniprocessor architectures, has resulted in a major research impetus to develop a new generation of parallel systems. Similar to a number of electronic symbolic computers being developed, optical computing systems can be viewed as a representative of the class of fine-grained, tightly-coupled architectures. This paper addresses potential roles of optical systems in symbolic computing and suggests future optical implementations of these architectures.

INTRODUCTION

The need to drastically increase the processing rate of artificial intelligence (AI) systems, coupled with the limitations of current uniprocessor architectures, has resulted in a major research impetus to explore parallelism in modern computing systems. Applications of these AI systems are placing stringent demands upon the underlying computer architectures. To meet these challenges, a new class of parallel computer architectures are emerging, structures which seek to optimize the processing and retrieval of symbolic information.

One promising group of AI-driven architectures can be described as tightly-coupled, fine-grained multiprocessor systems, and are characterized by both a dependence on complex interprocessor communications and flexibility in the interconnect topology.[1] This means that the architectures are composed of a large number of similar, relatively noncomplex processors,[2] which are coupled in such a manner that given processor can communicate with any other. The driving features behind this new generation of computer architectures are equally applicable to optical computing systems, which are also characterized by tightly coupled, fine-grained elements with a high degree of communications flexibility. Optical systems also provide a form of two-dimensional parallelism which appears are attractive for symbolic processing operations. At the present time, we believe that optical systems can be adapted to the types of operations and data structures encounted in AI, and offer the promise of enhanced computational throughput to overcome bottlenecks in existing AI systems.

This paper will first describe some of the aspects of symbolic computing which drive the need for these new architectures. As a case in point, opto-electronic interconnnects are already being investigated for interprocessor communications in computer architectures. Potential roles for optics in multiprocessor systems will then be addressed. The paper will conclude with a discussion of current efforts to utilize optics in symbolic computing, both at the interconnect and processor levels, along with proposed optical implementations which look promising for general purpose symbolic processing.

MULTIPROCESSOR SYSTEMS AND SYMBOLIC COMPUTING

Symbolic processing is typically refers to on going research and development in four functional areas: speech recognition, vision or image understanding, natural language understanding, and expert systems. All four disciplines are characterized by the following three attributes: symbolic representations of knowledge; a high degree of interaction with a stored grouping of symbolic knowledge; and some level of reasoning capability, which draws conclusions by comparing inputs to the system with elements retrieved from the knowledge base. At the present time, each of these factors, retrieval of knowledge, representation of knowledge and reasoning, limits the computational throughput rate of AI systems. The remainder of this section will use expert systems as an example of a symbolic processing domain.

An expert system is a machine with mimics or emulates the thought and reasoning processes

of human expert. It seeks to utilize the solution techniques which a human expert in a given discipline would use to solve particular problem in that domain. Knowledge appropriate to the discipline is placed into the machine, forming what is known as the knowledge base, enabling the system to understand the problem. Besides this knowledge base, the expert system consists of the "reasoner" or "inference engine", which manipulates the symbolic information stored in the knowledge base so as to "infer" new knowledge on its road to deriving a conclusion to a particular problem.

Retrieval in expert systems may be considered at four hierarchical levels: search of unorganized data, search of organized data, content addressing, and heuristic searching. The amount of symbolic information stored in a practical knowledge base is too large to even consider an exhaustive search of unorganized data. Considerable improvement can be realized by organizing the data to form data bases, and current expert systems make extensive use of data base management techniques to facilitate the retrieval process. But this technology is still based on von Neumann architectures which greatly hinder both the management and access to data bases of the size that will be needed for expert systems of the future. This problem has led to considerable interest in content addressable memory implementations of artificial neural systems and in logic-enhanced, or smart, memories for accomplishing heuristic searches.

Both the artificial neural systems and the logicenhanced memories fall into the category of tightly-coupled, fine-grained architectures. That is, they consist of thousands (or even millions) of switching or processing nodes (finegrained) which are interconnected to a high degree (tightly-coupled). The nodes of the neural network are just memory elements, but these elements are interconnected in a global fashion, so that processing associated with data retrieval may be distributed over all of the interconnected nodes. Although such networks will likely see use for symbolic computing, especially for associative recall of knowledge base information, neither the individual nodes nor the networks themselves have the processing power needed for advanced expert systems. Architectures with actual processing elements at each node represent the latest thinking of computer scientists dealing with artificial intelligence. These systems have been labeled with such names as logic-enhanced memories, smart memories, and connection machines.

Logic-enhanced memories avoid the von Neumann bottleneck by intermixing the processing and the memory funtions. This can be viewed either as distributing the memory among a large number of tightly-coupled processors, or as providing some processing capability to each element of a memory (hence the name logic-enhanced memory). This permits such powerful functions as interconnect reconfiguration and internodal relationship designation (e.g., for semantic network representations - to be discussed). These memories have such a broad processing power that they may be categorized as fine-grained, tightly-coupled multiprocessors.

Both the neural networks and the logic-enhanced memories are parallel processors; that is, any number of nodes or any of the interconnects can be active at any given time. It should also be noted that several optical implementations of neural networks and logic-enhanced memories have been developed, including associative processors[3,4] and memories incorporating a feature known as attention.[5] These systems all take advantage of the fine-grained nature of the optical devices and the types of global communications operations which are possible in optical computing systems.

A classification of parallel systems has been developed by Seitz,[6] a modified version of which is shown in figure 1. It provides an interesting categorization based on the number of processors and the relative degreee of processor complexity. Conventional uniprocessor architectures are plotted as a point of reference representing high complexity in a single processor. As one moves up to more than one processor, the trend is toward reduced complexity within each processor, a trend that is driven by total system cost and reliability, on the one hand, and by an escalating overall system complexity on the other hand.

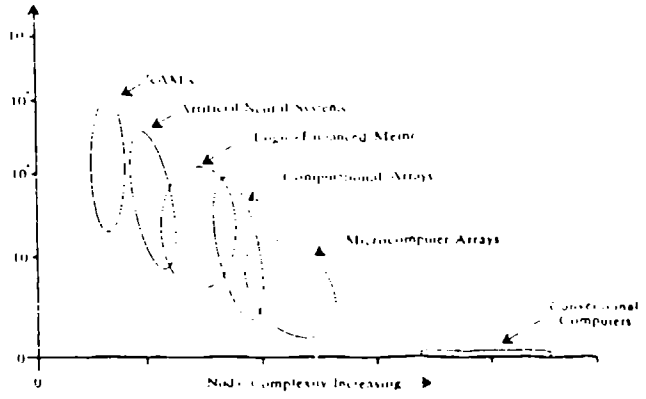


Figure 1
Classification of Parallel Processors by Nodal Complexity

Microcomputer arrays are basically a set of computers that send messages to one another via a communication network. Such systems are usually loosely-coupled; that is, the individual computers do not share main memory and i/O

devices, although one computer can always draw upon another's resources through the communication network. The application of such systems in symbolic computing will likely be in solving problems that involve interactions between more than than one knowledge base. Each processor can work on a given part of the problem in such a way as to minimize the need for interprocessor communications.

Computational arrays are systems whose processing elements have been designed for tasks of comparable complexity to floating-point operations. Systolic arrays, for which the processors are connected in regular patterns that match the flow of data in the computations, compromise most of the architectures in this category.

As mentioned above, the architectures of most interest to knowledge base retrieval are the more fine-grained, tightly-coupled systems, such as logic-enhanced memories and neural networks.

Random access memories are shown in Figure 1 as a point of reference on the fine-grained ends just as uniprocessors were shown as a reference for nodal complexity.

To this point the discussion has centered on the knowledge base retrieval process so fundamental to expert system operation. The second major aspect of expert systems, as mentioned above,is knowledge representation. Figure 2 illustrates one popular method for representing knowledge, known as a semantic network. A semantic net may be characterized as a graphical representation scheme in which the graph nodes represent objects or concepts and the links represent inference procedures that relate the nodes. The importance and complexity of connectivity in these knowledge representation schemes has resulted in serious consideration of the tightly-coupled multiprocessor architectures for symbolic computing and expert systems implementations. The processing power at each node can, for example, be used to define the internodal relationships of the semantic network.

The similarity of these highly connected architectures to neurological systems lends credence to their importance in symbolic processing. We are very aware of the power of the brain in performing intelligent operations such as reasoning and pattern recognition, yet the brain consists of relatively slow switching elements. The biological switch is the neuron, and it operates in the millisecond range - about a million times slower than current electronic switching speeds. The difference lies in the large degreee of connectivity between biological switches, leading to a high degree of parallel processing. Neurons in the brain can have upwards of 10,000 synapses (biological connectors), whereas electronic switches in today's computers typically have only a few connections to other switches. There is strong evidence to suggest that the processing power of the brain is related to the high degree of connectivity between the neurons, permitting parallel processing and thereby compensating for the slow switching speeds.

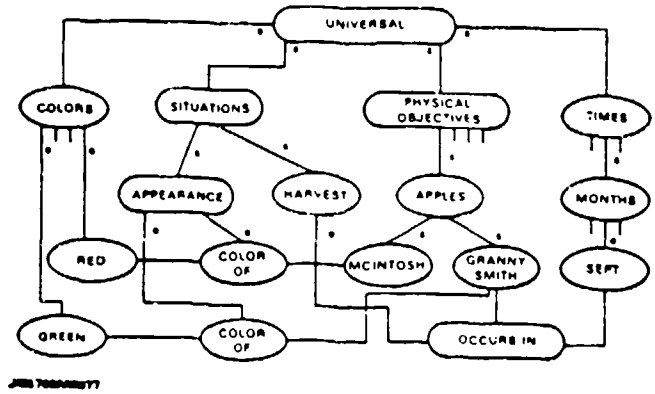


Figure 2
Semantic Networks

The next section will focus on potential roles for optics in multiprocessor systems. This will lead to a discussion of current efforts to couple optical computing with AI, and will conclude with two proposals for optical architectures which could significantly enhance the computational throughtput of fine-grained, tightlycoupled, multiprocessor systems. Such opticallybased systems, if realized, could open up a whole new field of opto-electronic computing directed toward artificial intelligence applications.

OPTICS AND MULTIPROCESSORS

Any approach to optical architectures must give serious consideration to what can be accomplished with existing technologies, namely VLSI electronics.
Optical computing will not seriously threaten electronic computing unless it can offer several orders of magnitude improvement in some critical measurement criterion, such as the power-speed-cost product, in a given problem domain. Therefore, a good starting point in addressing the application of optics to symbolic computing is to identify problem areas for electronics.

It is not surprising that the relative weaknesses and strengths of electronics and optics are traceable in one way or another to the fundamental physics of inter-electon and inter-photon interactions. Relatively speaking, the interaction between photons is weak; hence, electrons are good for the switching operations so fundamental

to computing and photons are good for the inter-switch communications, providing links which are free from detrimental coupling effects that lead to crosstalk and capacitive loading. Subscribing to such reasoning, however, is impractical due to the quantum losses which accompany both the electron-to-photon and the photon-to electron conversions. There is research and development underway to replace some of the longer interconnect links within computers with optical channels because it is the longer interconnects that create severe power, speed, and space problems for electronics[7]. But such a capability stops far short of using optics to its full advantage in multiprocessor architectures appropriate for symbolic computing.

Consider the electronic-switching/optical-communications positions as representing one of the four corners of the square shown in Figure 3. The sides of the square represent a continuum of combinations between the extremes of the corners. The upper left corner represents all-electronic systems while the bottom right represents all-optical. Since movement toward the bottom left corner not technically practical, the focus is along the upper and right sides. Upon considering computing systems for which switching is the predominant function, the trade-off between optics and electronics is seen to fall somewhere along the upper edge; that is, all electronic switching with some optical links. However, symbolic processing places a strong emphasis on connectively as was discussed above. The de-emphasis on switching (fine-grained architectures with low nodal complexity) and the emphasis on communications (tightly-coupled systems) leads one to consider architectures for which the communications is optics and only some of the switching is done with electronics. It is this category of electronic/optical hybrid architectures that can have a significant impact on symbolic computing.

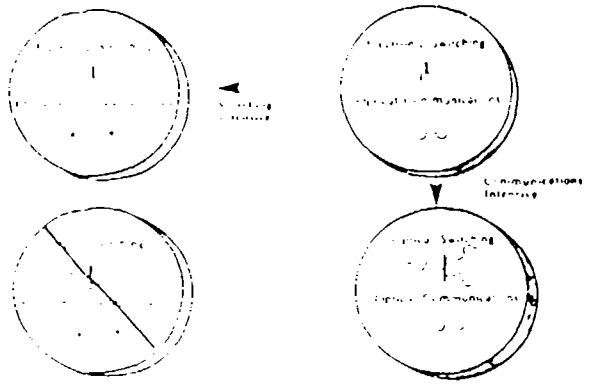


Figure 3
Electronic versus Optical Computing

The upper right hand corner of figure 3 refers to the set of architectures which utilize optical switching for interconnect reconfiguration and eletronic switching for logic operations.[7] Optics proves to be especially valuable in providing both the longer and the more global interconnects in processing, due to the combined power-speed-space-crosstalk penalties associated with electronic interconnects. Several examples of the use of optical interconnects for these in advanced architectures are currently under development, both in coarse and fine-grained computer structures. The interest in optics arises from the high bandwidth communications requirements between the individual nodes and the number of parallel channels which can effectively be multiplexed over a single optical link. The Texas Reconfigurable Array Computer (TRAC),[8] for example,is studying fiber optics for internodal board to board communications at bandwidths between 100 and 500 Mbps, and the WARP system[9] is investigating the use of optical interconnects for intercell communications at a rate of 0.5 1.0 Gbps and 32:1 multiplexing. Another example, that of a finegrained electronic symbolic computer under development, is the Connection Machine.[2] It is composed of 65,536 individual processing elements (PEs), organized as a large array of printed circuit boards, each of which contain 512 PEs equally divided between 32 chips. Both guided and unguided optical interconnects are being studied for overcoming challenges to this architecture, such as clock skew, 1.0 - 3.0 Gbps communication rates over 32:1 multiplexed links, and inter-processor broadcast.

Over the longer term, architectures such as that shown in Figure 4 could be developed to effectively integrate opto-electronic components in computer systems. Such a hybrid structure would use optics for most communications requirements and electronics for processing element functions. For the sake of simplicity, the illustration shows only two of the many possible boards and only four chips/board. If this were a fine-grained processor, each chip could contain many PEs.

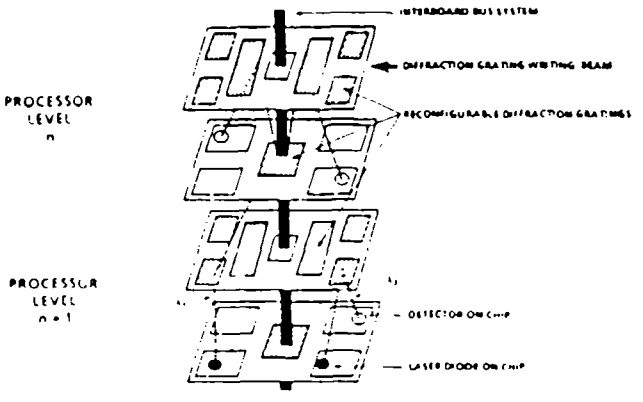


Figure 4
Hybrid Optical/Electronic
Multiprocessor Architecture

Each board in Figure 4 contrains four optoelectronic chips and one frequency selective filter (hologram). In between each board is a planar array of reconfigurable diffraction gratings, which perfom the majority of the switching operations involved in the interconnection process. This particular arcnitecture employs wavelength division multiplexing (WDM) to direct optical bit streams to the appropriate board. The beam labeled illustrates this operation. The hologram directly above of the transmitting chip directs the beam to the center of the next board, where it is superimposed on the main beam which travels to all of the systems boards. Upon reaching the intended board, the frequency selective filter diffracts the beam to a bus-to-board hologram which directs the beam to its final destination.

The intra-board and intra-chip interconnects would be handled by the plane of holograms above the board, as illustrated by beam . The logistics of handling a large number of muliplexed beams will not be discussed here, other than to say that the optical switching most likely will be achieved through nonlinear wave mixing. For example, four wave mixing may be used to generate holograms[10] which can be rapibly varied to permit interconnect reconfiguration. Diffraction grating writing beams would contain the desir d information for changing the holographic gratings. Note that some of the switching actions of such an architecture are being performed optically rather than electronically.

As one moves toward the bottom right corner of the classification scheme presented in Figure 3, the percentage of optical implementation increases until an all-optical architecture is achieved. While a number of efforts are underway to develop such all-optical structures, the research is currently directed at defining the appropriate computational primitives for optical symbolic processing. Thus the present focus is constrained to identifying and protoyping systems which can process the fundamental operations associated with symbolic computing--namely, the performance of correlation, searching, and pattern matching operations on symbolic data.

Typical of a class of research applications are the optical inference machines.[11,12] Here, the emphasis is on developing optoelectronic architectures which can perform the fundamental matching and logic operations encountered in retrieving symbolic data from a database. These sets of operations are both language and representation specific, so that architectures are being analyzed for several of the major different AI programming paradigms. Examples of other representations being investigated by optical computing researchers include semantic networks, graph theoretical representations, symbolic substitution for binary data,[13] and shadowcasting structures.[14]

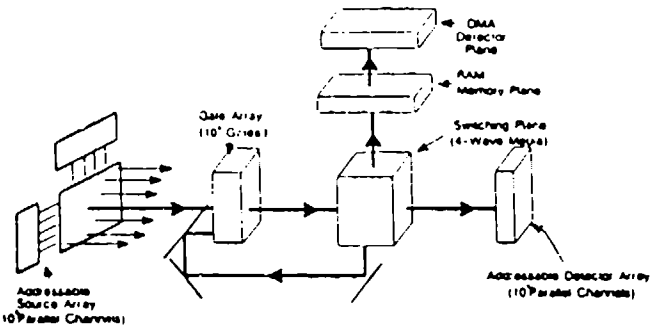


Figure 5
All-Optical Multiprocessor Architecture

An example of a fine-grained, tightly-coupled optical symbolic computer of the future is shown schematically in figure 5.[7] Although no one has built such a computer, it is technically believable to achieve such a system consisting of 1 million parallel channels. This does not mean that the system would be configured necessarily with 1 million nodes, since such this implies that the planar array of logic elements (designated as the gate array) would have just one logic element per channel. Instead, several logic elements would usually be interconnected via the interconnect media to form a processing element. For example a square array of n x n logic elements (gates) may comprise an arithmetic logic unit, several registers, and possibly some cache memory. An example of this type of structure is shown in figure 6, where individual elements in a 2-D SLM have been assigned the necessary functions to comprise a computational processing element. Taking an n of 5 (25 logic elements/processor) would lead to a machine with 40,000 nodes large enough to be practical as a symbolic computer.

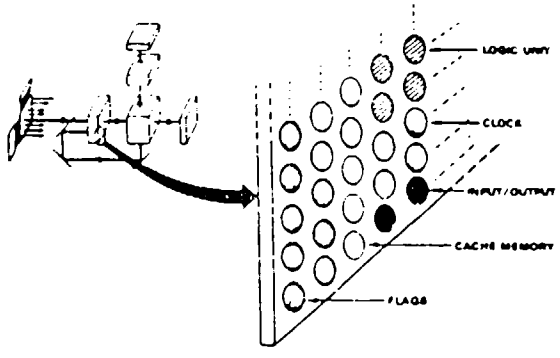


Figure 6
An All-Optical Processing Element
(Detail of Figure 5)

input to the optical computer could be via either an array of independently addressable laser diodes or a two-dimensional spatial light modulator (2D SLM). The diode array would be capable of much higher modulation speeds, but would involve more complex circuitry, especially if operation requires uniformity over the complete array. If the input already exists as a two dimensional light pattern, such as might be output from a vision processor, an input device may not be needed (depending on the compatiblility of the two processors).

The logic element array could be either a 2D SLM exhibiting a nonlinear response or an array of optical bistable switches. The latter device will ultimately lead to much higher switching speeds, but current realizations of optical bistable switches require impractical power levels. Improved nonlinear optical materials are needed to achieve widely utilized optical bistable devices.

The interconnect element will likely employ wave mixing in a nonlinear optical medium, similar in operation to that mentioned previously fo the hybrid architecture. However, due to the much larger number of channels that must be handled, the switching may be done in a multistage fashion, in which multiple parallel planes of real-time hologram arrays would be exercised.

The detector will be a major technological challenge. In the most general case, one would like a one million channel device, with each channel operating around 1 MHz (projected speed for 2D SLMs). However, the requirements will be much less for most practical processor designs. If the problem domain were to require, say, 100 iterations or more (e.g., semantic network searches to depths of at least 100), an output would be required only once every 100 microseconds. This reduces the throughput rate of the detector to 10^{10}, a number more in line with projections for GaAs microelectronics. Another example would be where each processor consists of a block of n x n channels as discussed above. Assuming an n equal to 4 and that each processor has just one output channel, the throughput requirement of the detector would be 6.25×10^{10}. Some combination of these two designs should yield a detector requirement that would be well within near technical feasibility.

The last major component of this all-optical architecture is the memory. The practice in electronics of co-locating some of the memory with the logic elements cannot necessarily be transferred to the optical computing domain because of the greatly reduced communications delays. Thus, Figure 5 shows the main memory as the single block, equally shared by all of the processors.

CONCLUSION

The ultimate objective of artificial intelligence is to expand the power and reasoning processes in computing machines, allowing these machines to emulate and achieve capabilities typically associated with intelligent behavior in humans. However, efforts to achieve these goals have been severely constrained by today's serial architectures and by the separation of the processing and memory functions. Fine-grained, tightly-coupled multiprocessors appear to be a viable class of architectures for symbolic processing. Optical techniques, in the form of opto-electronic interconnects and optical computing, are being investigated as a means of alleviating computational bottlenecks in these systems.

Opto-electronics may play a major role in making these systems a reality, either as a supplement to an existing architecture or as part of an integrated opto-electronic computer. Research to date has demonstrated the viability of optical interconnects, and a number of efforts are underway to incorporate opto-electronics into existing architectures. At another level, the potential exists for developing all-optical symbolic processors as part of a larger scale computational environment. This appears particularly promising, since the 2D SLMs are in fact fine-grained PEs in a tightly-coupled environment. Such an all-optical system could serve as a co-processor in symbolic processing systems, and research is underway to identify the appropriate computational primitives and compatible representations.

Interestingly enough, at the present time, the numeric "supercomputers" execute AI functions at a greater rate than dedicated AI machines.[15] This may imply that raw speed is an important component in overcoming existing AI computational bottlenecks. It also may indicate that architectures designed to improve numeric throughput may also be useful in symbolic computation, and vice versa. Regardless of which of these paths are developed, the utilization of optics in symbolic processing represents an exciting new direction for optical computing.

REFERENCES

(1) A. Gupta, C. Forgy, A. Newell, and R. Wedig, "Parallel Algorithms and Architectures for Rule Based Systems," Proceedings of the 13th Annual International Symposium on Computer Architecture, (1986)

(2) W. D. Hillis, The Connection Machine, MIT Press, Cambridge, MA (1985)

(3) A. D. Fisher, C. L. Giles, and J. N. Lee, "Associative Processing Architectures for Optical Computing," J.Opt.Soc.Am.A. 1 (1984) 133

(4) D. Psaltis and N. Farhat, "Optical Information Processing Models of Neural Networks with Thresholding and Feedback," Optics Letters Vol 10 (1985) 98

(5) R. A. Athale, C. B. Friedlander, and B. G. Kushner, "Attentive Associative Architectures and Their Implications to Optical Computing," Proceedings of the SPIE, Vol. 625, (1986)

(6) C. L. Seitz, "Concurrent VLSI Architectures," IEEE Transactions on Computers, Vol. C-33.12 (1984)

(7) J. A. Neff and B. G. Kushner, "Optics and Symbolic Computing," in Optical and Symbolic Computing, R. Arratnoon, Ed., Marcel-Dekker, New York (to be published January 1987)

(8) G. J. Lipovsky, Texas Reconfigurable Array Computer, University of Texas Technical Report, (1984)

(9) E. Anould, T. Gross, H. T. Kung, M. S. Lam, O. Menzilcioglu, K. Sarocky, and J. A. Webb, "WARP Architecture and Implementation," Proceedings of the 13th Annual Interantional Symposium on Computer Architecture, (May 1986)

(10) D. M. Pepper, "Nonlinear Optical Phase Conjugation," Optical Engineering, 21, (1982) 156

(11) G. Eichmann and H. J. Caulfield, "Optical Learning (Inference) Machines," Applied Optics, Vol. 24.14 (1985) 2051

(12) C. Warde and J. Kottas, "Hybrid Optical Inference Machines: Architectural Considerations," Applied Optics, Vol. 25.6 (1985) 940

(13) K. Brenneer and A. Huang, "An Optical Processor Based on Symbolic Subtitution, " Technical Digest of OSA Topical Meeting on Optical Computing, (March 1985)

(14) J. Tanida and Y. Ichioka, "Optical Logic Array Processor," Proceeding of the 10th International Optical Computing Conference, (1983)

(15) R. P. Gabriel, Performance and Evaluation of LISP Systems, MIT Press, Cambridge, MA(1985)

IMAGE UNDERSTANDING BY COMPUTER

EXECUTIVE SUMMARY

Image understanding is the process by which a computer interprets a scene. Significant progress has been made over the last few years on building computer vision systems. Systems have been developed that operate on complex real imagery in such tasks as medical diagnosis, inspection of industrial parts, and interpretation of remotely sensed data. Most of these systems are designed to be special purpose, with excessive domain-specific constraints. These domain specific designs make development of new systems very time-consuming and expensive. As a result, there is a clear need to develop image understanding systems that can deal with complex imagery that is also robust enough to be extended to other domains.

The development of these general purpose vision systems has proved difficult and complex. Research in the last few years, however, has opened the door to an array of computational theories of vision that will prove helpful in outlining processes that have application over a wide range of domains. Many of these theories have been prompted by studies on biological vision systems. This report will review a series of the computional processes that appear promising for designing a robust image understanding system.

The computational processes used in image understanding are of two general forms: numeric and symbolic. The numeric processing associated with computer vision is called low level processing. Low level vision may be characterized as a data representation of a scene in terms of sensor data without reference to knowledge of the content of the scene. A series of preprocessing operations are commonly performed at the low level prior to interpretation such as image formation and transformation, image restoration and enhancement, and image registration. Low level vision is also the level at which primitive yet rich descriptions of a scene are extracted. Some of these primitives are edges, lines, and regions. Image segmentation is also a part of low level processing.

Symbolic processing is known as high level processing. Understanding a scene entails far more than number crunching, rather, it involves representing image features in a symbolic form, similar to the way the human visual system processes information. The purpose of high level processing is to operate on intermediate representations derived from earlier vision processes in support of such functions as feature aggregation (perceptual grouping), object detection and recognition, scene interpretation, and activity and event description. These intermediate representations are in the form of symbolic representations. Processing over these symbolic forms is more concerned with global relationships, graph matching, and inference programing rather than some of the low level concerns including neighborhood processing, template matching, and numerical accuracy. The final objective of high level vision is to form a coherent form of the whole scene which combines the object information resident at the lower processing levels with previously stored information reflecting knowledge about the physical environment. This may involve contextually piecing together image features using syntactic (semantic) techniques, or through more heuristic methods.

The gap between these two levels is the intermediate level of processing. Intermediate level processing creates the symbolic forms that can be used for processing in the high level. Examples of symbolic forms involve describing an image in terms of regions and line segments as well as their associated attributes. The goal of intermediate level processing is to determine which features are relevant and the appropriate labeling of these features.

Implementation of image understanding algorithms is a key consideration for computer vision. Many of the computer vision systems require real-time or near real-time processing capabilities. Many of the traditional sequential von-Neumann architectures lack the processing capabilities necessary for many of the image understanding processes. The solution has been to implement the image understanding algorithms in a parallel environment. Many different parallel architectures have been prototyped to widen the computational bottleneck that results from the image understanding process.

Optical processing provides an alternative for solving the computational requirements in image understanding. The two principal advantages of optical computing are parallel processing, which maps nicely onto many standard image processing tasks, and its real-time operation. Optical computing offers a potential solution for processing over both the numeric and symbolic forms associated with image understanding. The general purpose optical computer is still some time away, requiring further development of logical gates, etc. There are, however, optical architectures that offer a reasonable near term solution to the computationally numeric processing requirements. These procedures are very competitive with conventional digital electronics for reasonably well-defined problems. Also, many of the operations performed in high level symbolic processing require high processing throughput at a relatively low degree of accuracy. Optics appears very amenable to this characteristic.

Whether optical computing can solve all problems in computer vision is still undetermined, however, many possibilities still exist. For example, alternatives exist which combine both digital and optical technologies. A hybrid digital/optical system can be realized, where front-end processing is performed by an optical system and intermediate to high level processing is performed by an intelligent back-end processing architecture. We should keep in mind that interfacing optical communication channels to electronic devices presents challenging technical problems. This report will analyze how well an optical parallel network can be used to solve the problem of image understanding in a computer vision system.

An introduction to image understanding is given in Chapter I. The three levels of processing which are involved in image understanding will be explained. We will also examine some of the inherent problems associated with algorithm formulation.

Chapter II examines the procedures used in early level processing. Some of the classic algorithms will be analyzed along with the rationale for using them. Low level processing basically involves preprocessing and creation of a primal sketch. Preprocessing operations compensate for

degradations resulting from sensor characteristics. The primal sketch is derived from a local neighborhood of image primitives. In many cases early level processing is a key step in the entire scene interpretation process since it lays the foundation by which further processing progresses.

Chapter III describes the role of segmentation in image understanding and examines the algorithms which have had success in many applications. Segmentation is the process of pixel classification, where the image is segmented into subsets by assigning the individual pixels to classes in order to generate a description of the scene. Texture and optical flow will also be examined as aiding in the segmentation process. Texture is the spatial distribution of pixels. Many techniques have been developed to describe this arrangement, and we will examine one that has had much success. Knowledge-based segmentation will also be explained along with some of the artificial intelligence (AI) techniques used in creating a knowledge-driven process. AI techniques are used for guiding and controlling the segmentation process.

Chapter IV looks at intermediate level processing for image understanding. We will step through some of the classification techniques that have been developed for pattern recognition, both statistical and syntactic. We will also describe the importance that intrinsic characteristics of an image has on image understanding. These intrinsic qualities are an important source of information for analyzing three dimensional scenes. Finally a powerful computational tool, called relaxation, will be examined. Relaxation is attractive since it is amenable to parallel implementation.

Chapter V explains the operations used in high level symbolic processing. Processing over symbolic representations of image elements provides the easiest way for a computer to understand a complex scene. Symbols supply information such as form, structure, and groupings whereas processing over symbols provide information such as relations, occurrences, and matching. Symbolic processing over image elements allows the vision system to make inferences about a scene, similar to the way a human would.

Many of the techniques used to process over these symbolic forms are based on artificial intelligence (AI) techniques. We will examine these AI techniques and provide a functional architecture dedicated to high level symbolic processing. Finally, we will briefly describe LISP, which is a popular symbolic data processing language.

In Chapter VI an example is given which demonstrates the techniques used in image understanding. In particular, the scene is an IR image of a tank. It starts at the low level stages and proceeds through the higher level processes. In this process the original image is transformed into a number of forms, each of which are more symbolic than the previous one. The techniques used to arrive at each form is explained , as well as the rationale for creating them.

Finally, Chapter VII summarizes the content of the report and provides a table that lists many of the important developments that have been reviewed. In addition, directions for further research are outlined.

Throughout the entire text, references are made to parallel implementations of the image understanding algorithms, both through digital and optical architectures.

CHAPTER I

INTRODUCTION

The processing requirements needed to solve the machine vision problem are not well understood. Presently no one can provide a detailed algorithm specification for a general vision interpretation system. However, it is possible to list the processing features which are necessary to significantly advance machine vision technology.

By machine vision, or image understanding by computer, we mean much more than image processing, which usually refers to the enhancement and/or restoration of images. Rather, the goal of machine vision is the automatic transformation of an image to a symbolic form that represents a description and understanding of the content of the image.

The image understanding process can be thought of as an iconic to symbolic (or signal to symbol) transformation. After performing such common image processing operations as edge detection and segmentation, which gives the "primal sketch" as proposed by Marr,[1] and grouping the primitives which result from these operations in terms of regions, contours and boundaries we are left with iconic features of the raw image data. To perform image interpretation the vision system must transform these iconic forms into symbolic form. The transformation is from a low level (e.g., pixel at location (20,100) has a red intensity value of 28) to a symbolic representation of the object in a scene, in terms of some predefined knowledge about objects in the world (that is, region 28 in the image corresponds to a particular object class TANK-GUN).

The machine vision problem can be described by three levels of processing - low, intermediate and high. The low level consists mainly of operations on pixels and local neighborhoods of pixels. This may involve segmentation algorithms to partition pixels into regions of similar color or texture properties, and/or extracting lines through intensity and color discontinuities at local edges. The result of this low level processing is a the transformation of a raw image to an image with regions and line segments.

The intermediate level of representation is an interface between the low level processing at the pixel level and the symbolic elements representing visual knowledge stored in a database. The intermediate level consists of a symbolic description of the two dimensional image in terms of regions and line segments as well as their associated attributes. Their are two general tasks for intermediate level processing. The first task entails the extraction of features for regions, lines, and vertices as well as relations between these elements. For example, regions supply information about intensity, texture, location, compactness, major axis orientation, etc. Line segments provide information on location, orientation, length, width, contrast, etc. Finally vertices convey location, connection of line segments, curvature, etc. The second task involves grouping, splitting, and labelling processes. These methods are used to form intermediate features which more naturally match stored object descriptions. Some operations are: (1) labeling points of high curvature on the perimeter of a region; (2) merging co-linear line segments; and (3) merging adjacent line segments.

The high level processing controls the intermediate level of processing where the symbolic two-dimensional representations of the intermediate level must be related to object descriptions stored in some knowledge base. That is, the objective of high level processing in image understanding is to operate on the intermediate representations (e.g., 2 1/2 D sketch, intrinsic images) derived from early vision processes in support of such functions as feature aggregation (perceptual grouping), object detection and recognition, scene interpretation, and activity and event description. In addition, high level processing can assist in focusing attention and resolving conflicts or uncertainty at low levels of the vision hierarchy.

In studying high level processing, two major classes of requirements predominate. The first class concerns knowledge representation whose requirements include the need to represent analogical, propositional and procedural knowledge, as well as iconic, geometric and relational structures. In addition it is desirable that the representations be easily

extensible, and facilitate rapid access and processing. The second class of requirements deals with processing, and includes the need to support data-driven and goal-driven processing, hierarchial and heteroarchial control, parallel and serial processing, and inquires into knowledge representations.

Part of this process entails control strategies for matching, merging and finally infering the presence of related objects. The result of this high level of processing is a symbolic representation of the content of a specific image in terms of a general stored knowledge of the object classes and the physical environment.

It is important to realize that the flow of information between all three levels of processing is in both directions. That is, the image understanding process can be data-directed (bottom-up), knowledge-directed (top-down), or both. In the upward direction, from the low level to higher levels, the communication consists of results from segmentation procedures that include a set of attributes of an extracted image event to be stored in a symbolic representation. The information allows processes at the higher levels to evaluate the success of the lower level operations. It is also a mechanism for passing actual symbols. In the downward direction, from the highest level to the lower level, the communication consists of commands for selecting subsets of images, specifying further processing, and requests for additional information.

A. IMAGE UNDERSTANDING PROCESSES AND ALGORITHMS

Image understanding is the process by which a computer is "programmed" to understand a scene. Recent developments in the area of image understanding have been fostered by the artificial intelligence community involved in machine vision. However, on a more rudimentary level, image understanding has developed through an extension of knowledge-based pattern recognition techniques applied to computer vision systems. Both approaches are based on human visual processes and Gestalt psychology, where experiments have shown that representing an image in a symbolic form

provides a more effective way to understand a scene, especially as a scene gets more complex. In this symbolic approach, a pattern/image is represented as a string, a tree, and/or a directional graph of pattern primitives and their relations. The decision making and/or structural analysis then becomes, in general, a parsing procedure.

There are some inherent differences between image understanding and pattern recognition that should be expounded. First, pattern recognition systems are concerned typically with recognizing the input from a small set of possibilities. Image understanding aims to construct rich descriptions of an image that can not be "programmed" in advance. Second, pattern recogniton systems are mostly concerned with two dimensional images, whereas image understanding systems operate mostly on three dimensional images. Finally, pattern recognition systems typically operate directly on the image. Image understanding systems operate on symbolic representations of the image that have been computed by earlier processes. It should be kept in mind, however, that these are not strict definitions.

One of the main problems today in machine vision has been the inability of researchers to clearly formulate those mathematical algorithms used for image understanding. Also, these algorithms have been so narrowly defined that their application to a generalized computer vision system is limited. Some of the promising prototype vision systems today which claim to be performing high level processing, from the 2 1/2 D to 3 D sketch, are riddled with heuristic and ad hoc procedures. That is, the processing that is used for the machine visison problem utilize schemes that follow some pre-defined rules that are only appropriate for a specific set of problems. The result is a set of algorithms that piecemeal techniques rather than a knowledge base system which evaluates scene primitives in a robust fashion. Consequently, whether the terms image understanding, symbolic computing, or pattern recognition are used to describe processess in an intelligent machine vision system, it is essential that one has a clear understanding of the algorithms. Once these algoritnms are understood it will be then easier to assess their utility for designing a reliable computer visison system.

The first process for image understanding is creating the 2D primal sketch. Many of the so called "low-level" pattern recognition and image processing techniques are used in this development. Next is the symbolic representation process which entails the grouping of features into symbols. Surface extraction can also take place in this process. Finally, there is the semantic or syntactic process which allows one to translate the symbols of the image into a description of a scene.

B. PARALLEL IMPLEMENTATIONS - OPTICAL PROCESSING

An important consideration in developing image understanding algorithms is how well they can be implemented. Many of the operations performed by understanding algorithms are computationally exhaustive and hence prohibit real-time implementation. The solution has been to implement these algorithms in parallel. Most of the approaches in the past have been for sequential implementation. This has resulted for the following reasons. First, the software has been designed for sequential applications. Second, it was not technically practical or even feasible to implement these algorithms with a parallel architecture. However, with the advent of VLSI and VHSIC design, parallel implementation of image understanding algorithms for machine vision has been possible.

The field of optical processing has grown to the point where real-time systems are being developed. The two principal advantages of optical computing are parallel processing, which maps nicely onto many standard image processing tasks, and its real time operation. This review will analyze how well a parallel network, such as optical processing, can be used to solve the problem of image understanding in a computer vision system. This correspondence will not attempt to prove that optical processing can solve the computational procedures but rather will refer to researchers who have reported success in using optical techniques in these areas.

The following paragraphs will look at the many ways in which a machine vision system can process and understand a scene in an image.

Then we will investigate whether this processing can be mapped onto a parallel processing architecture and/or an optical processing system. The approach will be to start at the "low-level" processes, refer to Figure 44 , and progress to the "higher" levels.

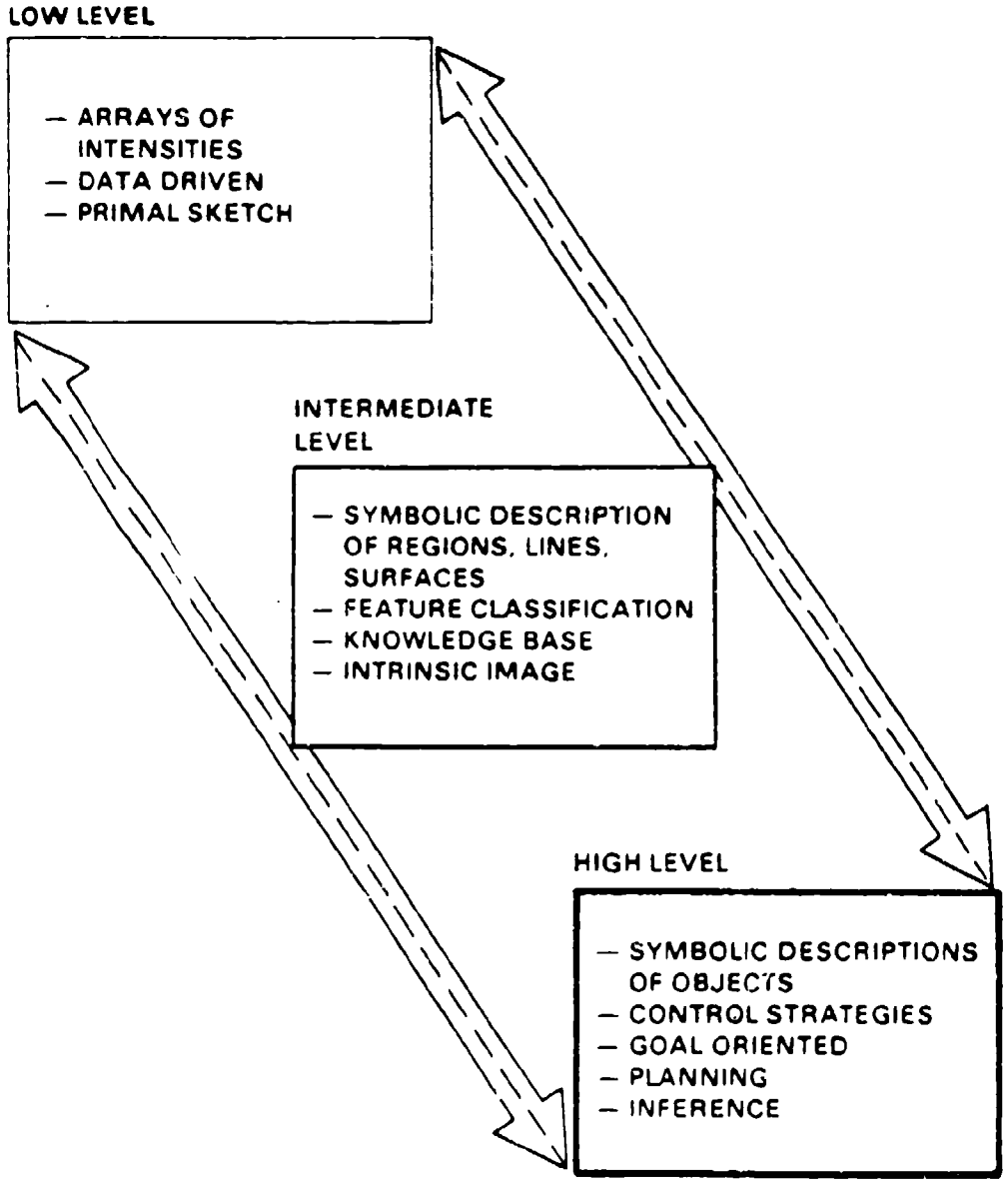


Figure 44. Communication, Control, and Representation in the Image Understanding Paradigm

CHAPTER II
PREPROCESSING AND THE "PRIMAL SKETCH"

The goal in image understanding is to create a representation of a scene that can be understood by a computer. However, as was mentioned before, this takes place over many levels of processing. The initial processing involves operating on the raw image data, which consists of a matrix of pixels that represent the gray level intensity of a scene. However, in most cases, the scene that is captured by the imaging system is not an ideal representation. Many standard image processing functions have been developed to compensate for this discrepancy.

One of the first operations performed in an image processing environment is to restore and/or enhance the image which has been degraded or deformed by the imaging system or the processing techniques used to recover that image. Some of the techniques used for restoration and/or enhancement are: (1) spatial filtering, used to accentuate certain characteristics of the image; (2) non-linear (median) filtering, used to suppress noise in the image; (3) transform processes, that isolate certain features of the image; (4) inverse filtering; (5) least squares (Wiener) filtering; (6) recursive filtering; (7) maximum a posterori (MAP); and (8) maximum entropy.

One of the computational methods used to perform the inverse filtering for image restoration is singular value decomposition (SVD). SVD is used to model the blurring impulse function (point spread function). If we represent the point spread function (PSF) P as a MxN matrix of rank R, then it is possible to decompose matrix P into a product of two unitary matrices and a diagonal matrix,

$$P = U \Lambda^{1/2} V^{T} \qquad (1)$$

where U is a MxM unitary matrix, N is a NxN unitary matrix and $\Lambda^{1/2}$ is an MxN matrix with a general diagonal entry $\lambda^{1/2}(j)$, called a singular value of P. It is possible to express Equation (1) in the series form,

$$P = \sum_{j=1}^{R} \lambda_{(j)}^{1/2} u_j v_j^T \qquad (2)$$

where the outer products, $u_j v_j^T$, of the eigenvectors form a set of unit rank matrices each of which is scaled by a corresponding singular value of P.

As mentioned before, image restoration attempts to restore an image that has been degraded. Using a general vector-space model to represent the transformation process from the input image, f, to the output image, g, as

$$g = Pf \qquad (3)$$

it is theoretically possible to recapture the ideal image, f, by finding the inverse of P. If P is properly characterized, then it is possible to use the concept of SVD to obtain the ideal image, f, by designing a filter that implements the P inverse operation, which can be expressed as,

$$P^{-1} = V \Lambda^{-1/2} U = \sum_{j=1}^{R} \lambda_{(j)}^{-1/2} v_j u_j^T \qquad (4)$$

The ideal image can then be expressed as,

$$f = P^{-1} g = V \Lambda^{-1/2} U^T g = \sum_{j=1}^{R} \lambda_{(j)}^{-1/2} v_j u_j^T g \qquad (5)$$

SVD has been very successful image restoration technique; however, it does suffer from being computationally intensive. The eigenvectors and must first be determined for the matrix, PP^T, and its transpose. Then the

vector computations must be performed. Even pseudoinverse computational algorithms that have been adapted for SVD are slow and are prone to severe ill-conditioning errors.

Parallel languages and architectures are being researched and developed to handle the preprocessing operations used in computer vision. The characteristics of the algorithms can determine which architecture is the most suitable. Two architectures that are appropriate are single instruction - multiple data (SIMD) and multiple instruction - multiple data (MIMD).

The general configuration of a SIMD architecture is depicted in Figure 45 . It consists of a single control unit, N processors, N memory modules, and an interconnection network (CN). Each processor is connected to its own memory module to form a processing element (PE). All processors execute the same instructions, hence, there is a single instruction stream. Each processor executes instructions on the set of data stored in its own memory module, hence there is a multiple data stream. The interconnection network facilitates communication between the processing elements.

The general configuration of a MIMD architecture is depicted in Figure 46 . It consists of N processors, a shared memory, and an interconnection network. Each processor executes its own set of instructions, hence there is a multiple instruction stream. Each processor also executes its instructions on the data in the shared memory, hence there is a multiple data stream. The processors access the shared memory through the interconection network.

SIMD architectures are generally more suitable for low level image processing where identical operations are performed on a large number of pixels. The image is partitioned into subimages and each processor is assigned a single subimage. Then, by executing the same set of instructions, all processors, in parallel, process all subimages of the image.

MIMD arcnitectures are generally more suitable for high level image analysis. The techniques involved in the higher levels of image understanding, such as the 2 1/2 D sketch, are pattern extraction and image

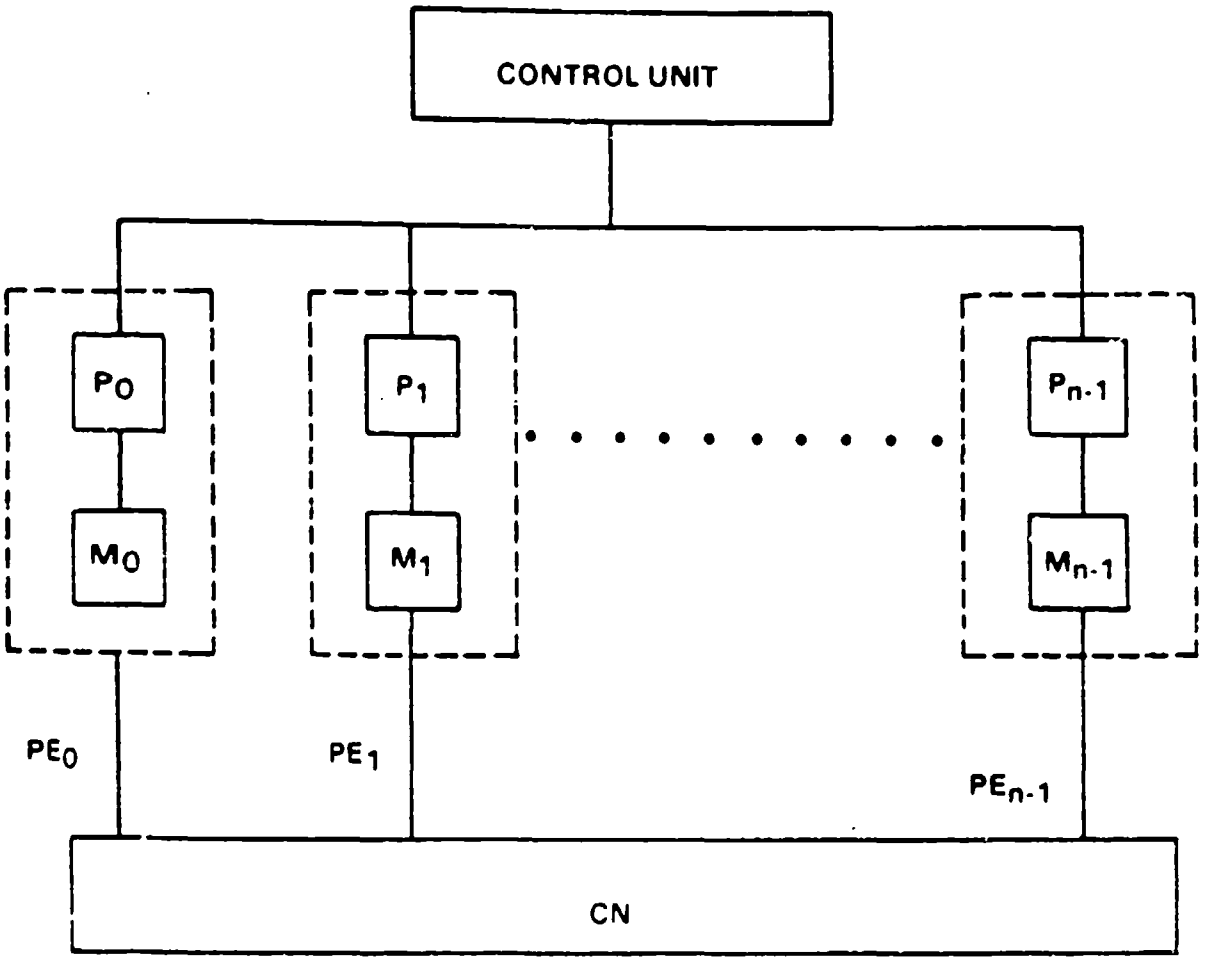


Figure 45. General SIMD Configuration

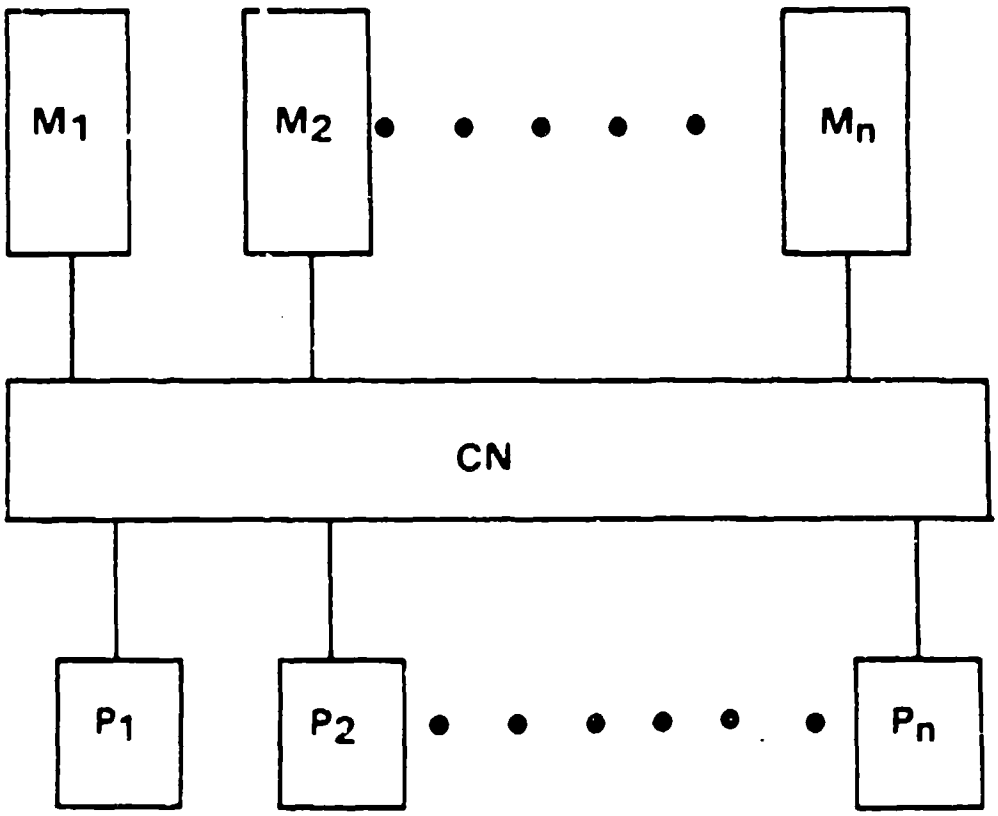


Figure 46. General MIMD Configuration

classification. Images at this stage are represented by data structures other than simple two dimensional arrays. The algorithms include many independent operations on common data that are well structured for MIMD architectures.

As mentioned earlier, many techniques used for image restoration and/or enhancement implement some form of discrete two-dimensional transform or solve a linear system of equations. Some examples of parallel implementations of these pre-processing operations follow. For example, a pipelined/parallel architecture has been developed for the two dimensional Fourier Transform (FFT).[2] Researchers[3] have implemented the least-mean-square (LMS) algorithm for image/signal restoration. Finally, others[4] have used a single instruction - multiple data (SIMD) architecture for image reconstruction.

Since SVD is based on outer product operations, it has been proposed that optical processors perform this matrix operation. For years researchers in optical processing[5] have promoted using optical processes for a variety of linear algebra operations. The use of an optical architecture for a SVD outer product calculation is very attractive for image restoration.

The next level of processing involves constructing a primitive but rich description of an image that can be used for further processing. The image processing and pattern recognition community have developed a variety of ways to recover these primitives. These primitives have been based on intensity changes found within an image, that is, an edge.

Many linear and nonlinear edge-enhancement and detection operators have been developed. However, studies have shown[6] that the nonlinear operators provide improved signal-to-noise (SNR) than such linear operators as high-pass filtering. A particularly attractive nonlinear edge-enhancement operator is the Sobel operator.

The Sobel operator, like other non-linear operators (Robert, Kirsch, Prewitt, etc.), utilize a non-linear combination of pixels as a means of edge enhancement. Most operate by processing over a 2x2 or 3x3 pixel window. The Sobel method operates by sliding a 3x3 window over an entire

image space. If we consider the pixel f(x,y) as the reference pixel, then by referring to Figure 47 , the edge value at e(x,y) is,

$$e(x,y) = \sqrt{X^2 + Y^2} \tag{6}$$

where,

$$X = (A2+2A3+A4) - (A0+2A7+A6) \tag{7a}$$

$$Y = (A0+2A1+A2) - (A6+2A5+A4) \tag{7b}$$

The angle, or the direction of the edge, at e(x,y) can also be calculated by,

$$\phi = \tan^{-1}(Y/X) \tag{8}$$

Thus Equation (6) and (7) describe the final image pixel value, e(x,y), as the nonlinear combination of its surrounding neighbors. It is also possible to describe the Sobel operator by the convolution of the 3x3 window with two linear local mask operators, X and Y, described now as,

$$X = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad ; \quad Y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{9}$$

Inspection of these mask operators in Equation (9) correspond to spatial differentiation (or since we are dealing with digital images, spatial differencing) with different weighting for pixels further from the mask center.

Since the Sobel operator is a local operation, as are the other edge detection operators, then parallel implementation is possible. Many

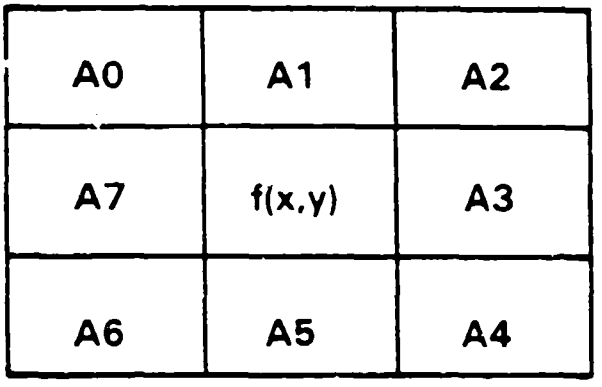


Figure 47. Numbering Convention for the Sobel Operator

parallel systems are being built to perform these local neighborhood operations. The general scheme is to design an architecture where individual processors are used to operate on individual neighborhoods using a SIMD framework.

The Geometric Arithmentic Parallel Processor (GAPP) system[7] is currently being used to perform local operations, such as the Sobel filter. The GAPP chip is an array of 72 parallel bit-serial processor elements that function in a systolic mode. The operations performed by the GAPP include convolving, sorting, and other arithmetic procedures.

In addition, optical systems can perform the linear spatial differentiation operation by implementing an optical matched spatial filter correlator. Briefly, the optical system's output is the convolution of the input image $f(x,y)$ and the reference function h (i.e., the impulse response of the system). When h is two delta functions separated by d, the system's output is

$$\text{output} = f * h = f(x,y) * \left[\delta(x+d,y) - \delta(x,y)\right] = \frac{\partial f}{\partial x} \quad (10)$$

or the first difference linear two-point approximation to the 1-D spatial differention of the input image $f(x,y)$.

By rewriting the Sobel output function from Equation (6) as,

$$g(m,n) = X^2 + Y^2 = (X + jY)^2 \triangleq |X + jY| \quad (11)$$

we are able to implement this complex arithmetic function optically so that the output amplitude at each point is,

$$\begin{aligned} |X + jY| = &((A2+2A3+A4)-(A0+2A7+A6)) \\ &+ j\,((A0+2A1+A2)-(A6+2A5+A4)) \end{aligned} \quad (12)$$

We can describe Equation (12) as in Equation (10) by the convolution of the input function $f(x,y)$ with a sum of delta functions at eight spatial locations with complex-valued weights for each delta function,

$$g(x,y) = f(x,y) * \Big[(1+j)\delta(x-d, y-d)+2\delta(x-d, y) + (1-j)\delta(x-d, y+d)+2j\delta(x, y-d) - 2j\delta(x, y+d)-(1-j)\delta(x+d, y-d) - 2\delta(x+d, y)-(1+j)\delta(x+d, y+d)\Big] \quad (13)$$

where d is the spacing between pixels in the input image and where the weights and locations of each delta function are obtained from Equation (12).

Researchers[8] have demonstrated that it is possible to realize both the desired impulse reponses and complex-valued weighting in Equation (13) optically. Specifically, the impulse response could be achieved by forming the holographic matched spatial filter (MSF) of an input function containing delta functions(apertures) at the correct locations and of the correct radii (to adjust the intensity) and with the necessary phase factors achieved by placing $\lambda/4$ and $\lambda/2$ plates behind the appropriate apertures. The complexed-valued weights for each delta function could be achieved by shifting the wedge in 1-D in its plane.

Another local edge operator that has been of interest, especially in the AI community, is the Laplacian of a Gaussian (D2G) edge finding technique. This operator has been used to find edges to create, what Marr has termed, the "primal sketch". Marr and Hildreth' reasoning for using this type of operator is to smooth the edge before taking the derivative, since derivatives are inately noisy. The smoothing is done by convolving the image with a Gaussian. Intensity changes, that characterizes edges, are found by locating the zero crossings produced by implementing the Laplacian, which is a non-directional linear second derivative operator. The D2G operator can be implemented by: (1) scaling the operator values by some constant; (2) using nearest integer values for each scaled operator value; (3) extending the support of the filter (window) to include all nonzero integer values; and (4) manipulating operator values by a small amount to ensure that the values integrate to zero.

It is interesting to note why a Gaussian was chosen as the optimal smoothing filter and the Laplacian was chosen for finding the edges. Two

physical considerations have combined to decide on using a Gaussian. The first is the need to reduce the range of scales over which intensity changes take place. This condition can be expressed by requring that the spatial frequency spread, $\Delta\omega$, be small. The second consideration is that the visual world is not constructed of primitives that extend over large areas, but rather of localized occurrences of pattern primitives, such as contours, shadows, creases, etc. This condition can be expressed by requiring that the spatial domain, Δx, be small.

Unfortunately, these two requirements are conflicting, since,

$$\Delta x = \frac{1}{\Delta\omega} \tag{14}$$

However, there is a relation, called the uncertainty principle, which states that $\Delta x \Delta\omega \geq 1/4\pi$. The distribution that optimizes this relation is the Gaussian. Thus, Marr and his predecessors have chosen the Gaussian as the optimal filter for image smoothing.

The reason for using the Laplacian is two-fold. First, an edge can be characterized as peak in the first directional derivative, or equivalently, a zero-crossing in the second directional derivative of intensity. The Laplacian is a second order derivative. Second, one needs an operator that is independent of the orientation. The Laplacian is the only orientation-independent second-order differential operator.

The difference-of-Gaussian (DOG) operator is an approximation to the Laplacian of a Gaussian convolved with the image. The DOG operator is created by the addition of one positive and one negative Gaussian-weighting function, with the variance of the two Gaussians having a ratio of about 1.6. The result of subtracting the two Gaussians to create the DOG operator is shown, in one dimension, in Figure 48 . The output of this difference resembles a mexican hat.

An application of the DOG operator for an intermediate vision understanding scenario is stero matching.[9] The algorithm consists of: (1) calculating the edges using the DOG operator at a coarse scale (i.e., where the window size or mask is large); (2) matching the edges in the

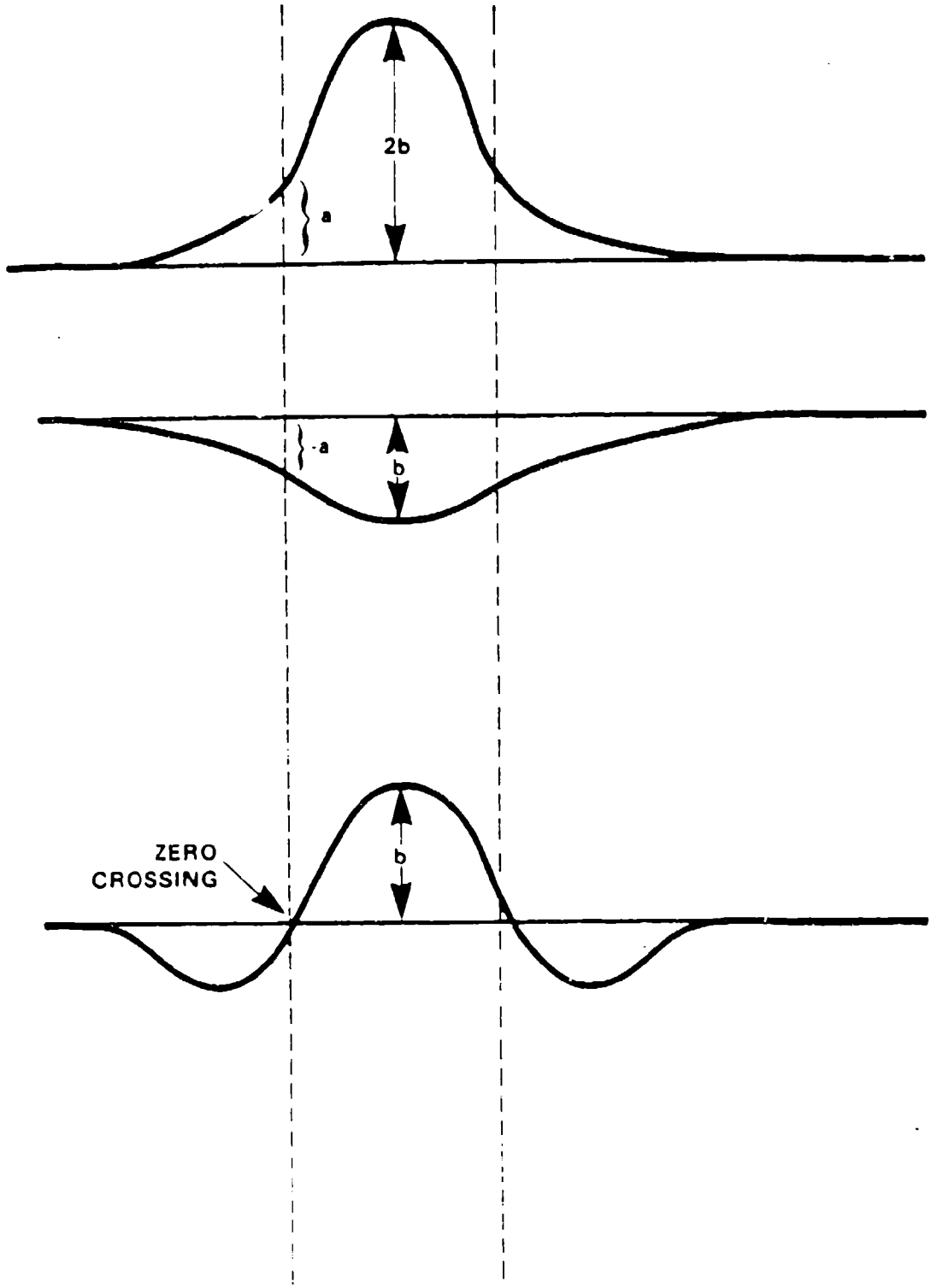


Figure 48. Creation of the DOG Edge Operator, "Mexican Hat"

binocular views; (3) calculate the edges at a less coarse scale; (4) match the edges, within the broad limits of the previous pass; and (5) iterate for finer resolution images, until complete.

Researchers[10] are presently in the process of constructing hardware to implement the DOG operator in a parallel fashion. More recently, an efficient optical imaging device that performs linear convolutions with two-dimensional circularly symmetric operators in parallel has been designed using VLSI technology.[11]

Very often the edges that are produced from implementing some local operator or edge fitting procedure also need some processing. For many man-made objects, edges correspond to the juxtaposition of surfaces or shadows. In a well focused image, edges should appear sharp and should extend in some direction for some length. Obviously, in the natural world, the boundaries are not necessarily as sharply defined, e.g., trees, fields, etc. However, the output of the operators described previously is generally smeared at or near the edge location. For many image understanding problems it is necessary to localize the edge so that it lies along the object boundaries. Given a knowledge of the edge detector, it is possible to design a process which accepts the output of the operator and produces a "thinned" representation of the edge at the location of maximum edge response to produce an edge map.

One of the more common techniques for edge thinning uses a process of non-maximum suppression. Non-maximum suppression is a process where edge magnitude values which are normal to the direction of the edge are suppressed. Obviously, it would be incorrect to consider simply those edge values of maximum response, since this would force adjacent points in the direction of the edge to compete. In practice, directional masks (north, south, east, west), among other similar techniques, are applied directly to the edge values to create the edge map. The algorithm proceeds by associating a directional mask with each edge point oriented normal to the maximum edge response at that point. The center point is then deleted (assigned zero response) if any point within the mask has a greater response. The edge masks are shown in Figure 49 . One of the

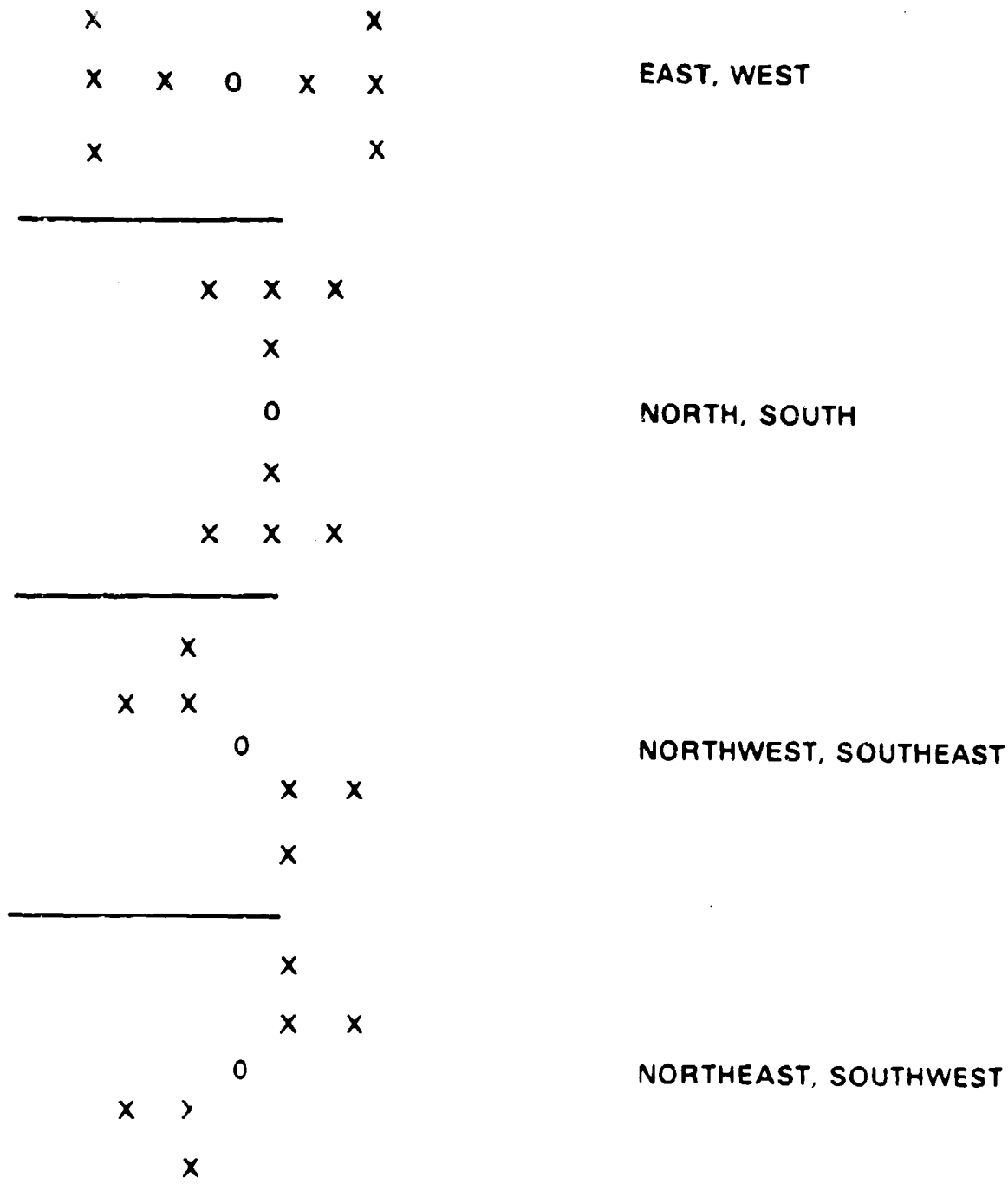


Figure 49. Edge Thinning Masks for Each Principal Edge Direction

main considerations for edge thinning is maintaining digital connectedness, whether it be 4-neighbor or 8-neighbor.

Since these edge masks perform local operations, they can be implemented in parallel. This thinning algorithm is best processed by an SIMD architecture. In this architecture the image would be divided among the processing elements on an equal size basis. Then each processing element would convolve the operator masks with the portion of the image it holds. The interconnection network would be used to transfer boundary pixels needed by neighboring processing elements.

Finally this edge thinning algorithm has the potential of being implemented in an optical processing environment, similar to the edge operators.

CHAPTER III
SEGMENTATION AND THE FULL PRIMAL SKETCH

The next step after we have produced the primitive characteristics of the scene is to find parts of the scene that belong to the same class, that is, we need to segment the image. Image segmentation is the process by which an image of a scene is broken into separate parts (regions or objects) in order to generate a description of that scene. Generally, segmentation is a process of pixel classification, that is, the picture is segmented into subsets by assigning the individual pixels to classes. Many different techniques have been developed for image segmentation, most being general purpose while others are dependent on the particular application.

Image segmentation falls into three general categories: (1) characteristic feature thresholding and clustering; (2) edge detection; and (3) region extraction. Examples of some of the segmentation techniques used include thresholding and spectral (color) signature classification,[12] region growing,[13] pyramid approaches (linking and spot detection),[14] double window filters,[15] border/edge followers,[16] relaxation,[17] superslice,[18] superspike,[19] etc. A more detailed explanation of these techniques can be found in an accompanying report.

Although these standard segmentation techniques are fairly widely applicable, it is not always obvious, given a class of images, which of them are applicable to that class. In fact, the success of a particular technique is dependent on how well an image satisfies a particular set of assumptions, which are not always explicitly stated. Therefore it is important that the assumptions (models) that underlie various basic image segmentation techniques are fully understood in order to successfully utilize them.

The standard method of segmenting an image is by thresholding. Here the classes correspond to grey level ranges, e.g. "light = hot" and "dark = cool". Since these ranges are not known in advance, they must be determined by examining the grey level histogram and looking for peaks

(one-dimensional clusters) and choosing thresholds (one-dimensional decision surfaces) that separate the peaks.

It is also possible to segment an image by thresholding the values of the local properties (other than the grey level) measured at each point. For example, suppose that a picture is composed of "busy" regions and "smooth" regions. It should be possible, in principle, to segment the image into these regions by computing some local measure of "busyness" at each point, and thresholding the values of this measure. Even better results can be obtained if we smooth the "busyness" values by locally averaging them over a neighborhood of each point since this tends to reduce variability of the values and hence make the histogram peaks easier to separate. The "busyness" values can be found by applying a difference operator over a neighborhood of each point.

Multidimensional thresholding has been used by Ohlander[12] for a segmentation scheme for natural color images. Ohlander's method is considered a recursive region splitting method. The algorithm can be described by referring to Figure 50 . The first step is to select a region of an image. The second step is to compute the histograms for all features for the portion of the image which is contained in the image. The histograms were representative of the red, green, blue tristimulus values, the television transmission tristimulus values (Y,I,Q) and a set of non-standard color coordinates called intensity, hue, and saturation. The third step is to select the "best" peak in the set of histograms. Fourth, threshold the image to a binary form using the upper and lower thresholds derived from the upper and lower bounds for the best peak in the set of histograms. Fifth, select the connected regions. Sixth, save these regions and check each region for further segmentations. Finally, continue the segmentation on the remainder of the region which was being segmented - terminate when there are too few points left.

Price[20] extended this method for segmenting monochromatic images. His procedure implemented two general modifications: (1) planning, used to improve the speed of the procedure, and (2) texture operators. The planning procedure[21] performs the segmentation on a reduced version of the

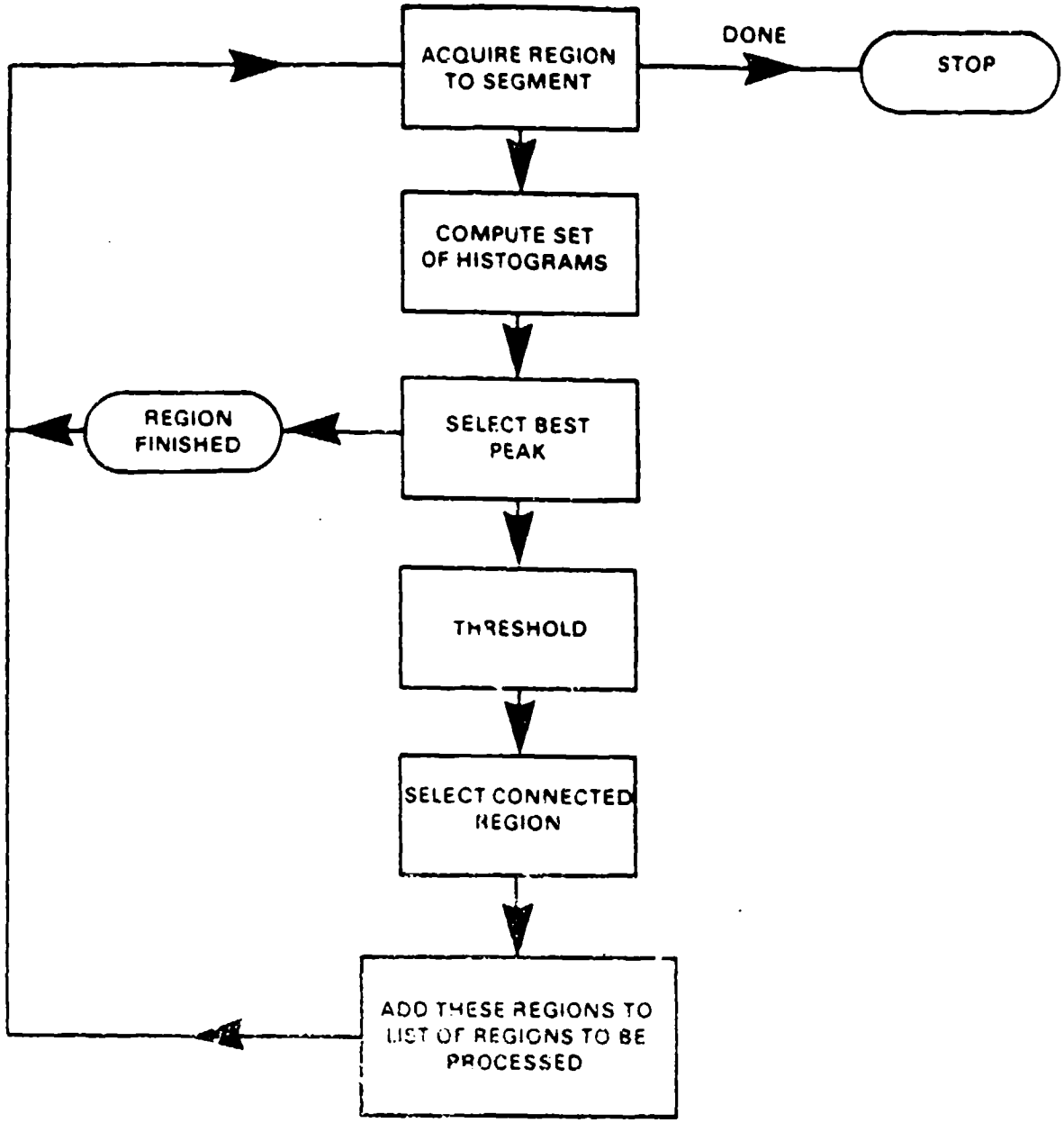


Figure 50. Segmentation by Recursive Region Splitting

image and uses this segmentation as a plan for the final segmentation of the full size image. The textural operators are added to generate more features for segmentation. They are based on edge operators.

Initial analysis has indicated that thresholding by cluster detection in histograms is not adequate for separating targets from background. Grey level target and background clusters are often not separable, i.e., their probability densities overlap. The basic disadvantage of segmentation schemes which use only local feature values is that they attempt to classify image parts without regard to their relative positions in the image. The segmentation techniques that are being implemented today, which are mentioned in an accompanying report, rely on processes that make use not only of similarity but also proximity.

Segmentation by thresholding is a common low level image understanding technique. There has been research into implementing this procedure in a parallel environment in order to increase the speed of computations. Two popular architectures, the WARP[22] and ZMOB,[23] have implemented a histogramming technique with thresholding successfully. Another successful architecture, PASM,[24] uses a partitionable SIMD/MIMD system for building histograms. The GAPP system has also been used to perform histogramming. Finally, researchers[25] have proposed ways to threshold in an optical processing environment using a microchannel spatial light modulator.

Segmentation based on an edge detection is very common. There are many algorithms that exist which capitalize on the information supplied by edge information. An algorithm that links edges based on characteristics of the imagery has been used successfully in a number of applications.[26] An extension of this technique has been used in an object recognition/classification mode where symbolic descriptions are created from knowledge about the scene. The algorithm is used to extract linear features in an image by a process of edge detection and derive higher level descriptions from the extracted lines.

The algorithm consists of the following steps. First, determine edge magnitude and direction by convolution of the image with a number of edge

masks. Second, thin and threshold the edge magnitudes. Third, link the edge elements based on proximity and orientation. Fourth, approximate linked elements by piecewise linear segments. The first two steps in the algorithm are similar to the techniques of edge detection and thinning that have already been discussed. The "linking" procedure is the main crux of the algorithm. The linking algorithm is based on a set of rules which have been derived both heuristically and through a general knowledge about edges. Using an eight-neighbor model for connectedness, neighbors of edge elements are determined by predecessors and/or successors. Rules are set up to decide whether a predecessor/succesor is found. The result is edge point configurations giving the "best" open boundary. Then an intermediate step is initiated to form boundary segments by operating on the predecessor/successor edge image. Finally an iterative end-point fit algorithm is implemented to form line segments for the most distinct boundaries.

A top-down knowledge about the scene can be used to create a symbolic description for certain classes of objects. The symbolic description is in the form of a logical rule base. For example, applying the edge linking algorithm to imagery that is known to contain roads and runways would define a rule base that searches for elongated parallel lines, where the lines have opposite contrasts. An additional search tool that would help in describing these elongated parallel lines is the width of the pair of lines and medial line information.

The edge linking algorithm performs predominantly local operations and thus could be implemented in a parallel architecture. An appropriate architecture is SIMD. Since many of the rules used in the linking algorithm can be programmed as logical comparisons then the linking can be performed in local neighborhoods by processing elements. It may be more appropriate to have the image divided among the processing elements on an equal "interest" basis rather than having the PE's divided on an equal size basis to maintain continuity in boundary formulation.

The third type of segmentation procedure is region growing. Region growing uses image characteristics to map individual pixels in an input

image to sets of pixels called regions. The most primitive region growers use only aggregates of properties of local groups of pixels to determine regions. More sophisticated techniques "grow" regions by merging more primitive regions.

Region growing techniques can be split into three different areas: (1) local, (2) global, and (3) splitting and merging. Local techniques involve placing pixels in a region based on their individual properties or properties of their close neighbors. Global techniques group regions on the basis of the properties of large numbers of pixel distributed throughout the image. The algorithm used by Ohlander is an example of a global procedure. Finally splitting and merging techniques operate on individual pixels and use state space procedures to merge or split regions using graph structures to represent regions or boundaries.

The splitting and merging technique proposed by Brice and Fennema,[13] and later extended by Feldman and Yakimovsky[27] and others,[28,29] has been an important development for segmentation since domain-dependent "semantics" were incorporated into the analysis. That is, knowledge about the scene was used to improve the performance of the segmentation procedure. Also, their algorithm promoted attempts to understand complex scenes using an artificial intelligence methodology.

In order to perform splitting and merging Brice and Fennema developed a boundary representation. This representation, refer to Figure 51 , describes the image in the form of two grids, the supergrid, S, and the image grid, G. The . and + represent the supergrid and o represents the sub-grid. The representation is assumed to be four-neighbor. Boundaries of regions are then defined at points marked +.

The region growing algorithm can be described in two parts. The first is the non-semantic algorithm. Proceed by merging regions i,j as long as they have one weak separating edge. A weak separating edge is defined heuristically with a threshold criteria. Then merge regions i,j when $S(i,j) \leq T1$ where,

$$S(i,j) = \frac{c_i + \alpha_{ij}}{c_i + \alpha_{ij}} \qquad (15)$$

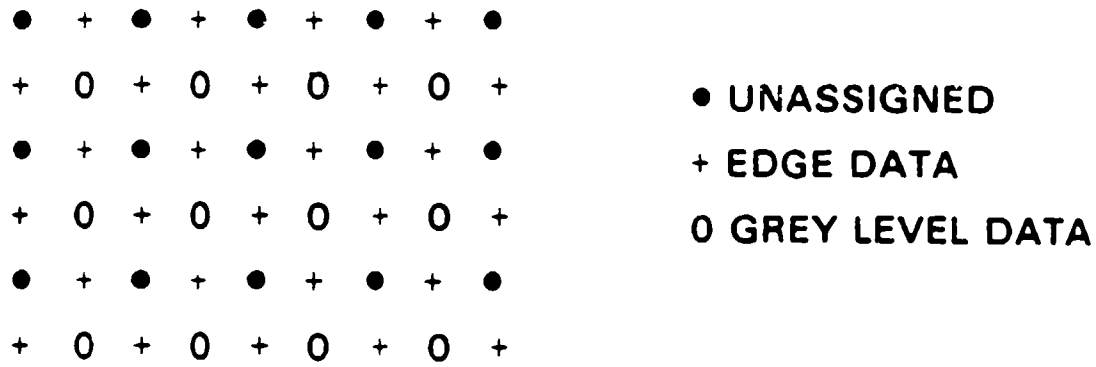


Figure 51. Grid Structure for Region Representation

where,

$$\alpha_{ij} = \frac{\sqrt{\text{AREA}_i} + \sqrt{\text{AREA}_j}}{(\text{PERIMETER}_i)(\text{PERIMETER}_j)} \qquad (16)$$

until no two regions pass this test.

The second part is the semantic algorithm. Let Bij be the boundary between Ri and Rj. Evaluate each Bij with a Bayesian decision function that measures the conditional probability that Bij separates the two regions Ri and Rj of the same "interpretation". Merge regions Ri and Rj if the conditional probability is less than some threshold. Then evaluate the "interpretation" of each region, Ri, with a Bayesian decision function. Assign the interpretation to the region with the highest confidence of correct interpretation. Update the conditional probabilities for different interpretations of neighbors.

The semantic part of this region growing algorithm maximizes an evaluation function that measures the probability of a correct interpretation, given the measurements on the boundaries and regions of the partition. A criteria function was derived to determine the probability that a boundary Bij between regions Ri and Rj is false. Finally, a confidence measure was derived as a ratio of conditional probabilities to determine the most likely interpretations.

Implementation of region growing in a parallel environment is feasible since many of the operations are performed on a local neighbornood of pixels. The difficulty would be in "programming" the semantic criteria, as proposed in the previous algorithm, in parallel.

A. TEXTURE AND OPTICAL FLOW

Two characteristics of images that has become increasingly useful for segmentation are texture and motion (optical flow). Texture has been used

by many researchers for image understanding in scene analysis. Unfortunately, there has been as many defintions of what texture is as their has been researchers investigating it. Image texture can be qualitatively evaluated as having one or more of the properties of fineness, coarseness, smoothness, randomness, etc. Texture can be described by the number and types of texture primitives (texels) and the spatial organization of its primitives. The spatial organization may be random, may have dependence on one primitive, or may have a dependence of n primitives at a time. The dependence may be statistical or structural.

The structural models represent the texels as a repeating pattern and construct rules for generating them. The model is best suited for describing patterns which have texels that are highly regular. An example of a structural approach to texture analysis will be given in the syntactic pattern recognition section in Chapter IV.

The statistical models describes texture by statistical rules that govern the distribution and relation of grey levels. This technique is appropriate for images of natural scenes whose texels are hard to differentiate. Many of the techniques used in the statistical approach will also be referenced in Chapter IV. The general procedure is to create a set of features based on texels in an image for use in a classification scheme.

One of the statistical techniques that has had much success for texture analysis is the spatial grey level dependent (SGLD) co-occurrence matrix. The following will provide a general description of the co-occurrence matrix. Let (F) represent the discrete, quantized (MxN) image matrix. Let each element F(i,j) be a B-bit integer giving a range of $(0,2^B - 1)$. $P_{\alpha,\beta}(u,v)$ is denoted as the relative frequency with which two pixels, one with grey level, u, and one with grey level, v, separted by a distance, α, in the direction, β, occur in the image matrix $[F]$. For example, $P_{1,\beta}(u,v)$ is the directional joint grey level occurrence of adjacent pixels in a 3x3 neighborhood that are separated by distance one. Common practice is to restrict β to multiples of $\pi/4$. The directional matrices $[P_{\alpha,\beta}]$ describe the second order statistics of an image for a

given spatial direction, β, and separation, a. The co-occurence matrix can be used in histogram sharpening for segmentation. A set of features can then be calculated from the co-occurrence matrix, such as entropy, contrast, maximum probability, etc., for training in the classification procedure.

Parallel implementation of the co-occurrence matrix is possible because of the local nature of the operations being performed. A similar architecture using SIMD can be used to implement this algorithm, similar to process for the edge operators. Some differences would be in the manner in which the PE's are programmed.

The use of optical flow for segmentation has become increasingly popular over the last few years. Optical flow is the distribution of apparent velocities of irradiance patterns in a dynamic image. The velocity patterns and its discontinuities can be an important source of information about the arrangement and the motions of visible surfaces. When two adjacent surfaces undergo different rigid motions, a discontinuity in the the velocity field usually results along their boundary. Since a surface is continous across a crease, regions in the image on either side of a surface orientation discontiniuty exhibit consistent velocities, but the velocity gradient usually differs. In contrast, regions on either side of an occuluding boundary can have arbitrary velocities.

Horn and Schunck[30] recently suggested a technique for determining optical flow in the restricted case where the observed velocity of the image irradiance patterns can be attributed directly to the movement of surfaces in the scene, i.e., the irradiance at a point in the image is proportional to the reflectance of the surface at the corresponding point in the object (there is no shading). Under these circumstances, the relation between the change in the image irradiance at a point (x,y) in the image plane at time t and the motion of the irradiance pattern is given by the flow equation,

$$B_x u + B_y v + B_t = 0 \qquad (17)$$

where $B(x,y,t)$ is the image irradiance, and $u = dx/dt$ and $v = dy/dt$ are the optical flow components. By factoring in appropriate error terms and using the calculus of variations, Horn and Schunck derived an iterative solution, using a relaxation procedure, which resulted in the following equations,

$$u(k+1) = \bar{u}(k) - f_x \frac{L}{M} \tag{18a}$$

$$v(k+1) = \bar{v}(k) - f_y \frac{L}{M} \tag{18b}$$

where,

$$L = f_x\bar{u} + f_y\bar{v} + f_t \tag{19a}$$

$$M = \lambda^2 + f_x^2 + f_y^2 \tag{19b}$$

where f_x, f_y, f_t are the partial derivatives with respect to x,y, and t. $\bar{u}$ and $\bar{v}$ are the four-neighbor local average velocity components and λ is the Lagrange multiplier used in the smoothness constraint.

Parallel implementation of optical flow is possible since relaxation is clearly a parallel procedure. This is a parallel technique because the elements of the new approximation, $u^{(k+1)}$, may be computed simultaneously and in parallel by a network of processors whose inputs are elements of the old approximation, $u^{(k)}$. As such, it requires the storage of both the old and new approximations.

3. KNOWLEDGE-BASED SEGMENTATION

The quality of segmentation is crucial to effective machine understanding. Without the capability of good segmentation of physically meaningful regions, even the most intelligent processing cannot achieve satisfactory scene interpretation. In the past, the most useful segmentation results have been achieved using model driven techniques

which were tuned to work for specific situations. These segmentation methods work well for very focused applications such as finding bright targets in low clutter scenes. The performance of these segmentors goes down rapidly, however, when they have to deal with a wider range of imagery. There are many applications where computer vision systems have to function over a wide range of situations. For example, the computer system for an autonomous land vehicle must function in a variety of situations which will be affected by different sensors, terrains, weather conditions, and even the time of day. There are currently no segmentation methods that work well over this wide variety of situations.

A way to improve the process of segmentation and make it robust enough to handle a wide range of scenarios is to use Artificial Intelligence (AI) techniques to bring more knowledge to bear on the problem. Using AI cues, such as knowledge of the terrain or weather conditions, can guide and control the segmentation process. This knowledge can come from other sensors, the processing results from previous frames, or even pre-mission training.

Various knowledge driven image processing techniques have been studied. Duane et al.,[31] use knowledge driven production rules to evaluate the region segmentation of an image. Their evaluation is used to group smaller regions into larger ones. This evaluation can also resegment a region with new parameters. The segmentation is performed by a single routine that is data driven using no other knowledge. On the other extreme Nazif and Levine[32] use production rules to control all aspects of the segmentation process. Rules control the analysis and grouping of lines, and regions as well as the scheduling of different segmentation tasks.

An attractive approach to use knowledge effectively in the segmentation process is to incorporate knowledge driven algorithm modules. Each of the algorithm modules performs a specific step in the processing flow. To make a given unit knowledge driven, information that identifies or quantifies is included, such as knowledge about the type of sensor used, the time of day, weather conditions, or actual measurements from the

image. For example, an algorithm to do noise cleaning could use knowledge about the overall contrast to determine which operations to perform.

A knowledge base needs to be constructed to house all the information gathered. It can also be used to retain information that is acquired during the processing chain. In addition, a method is required to control the information in the knowledge base. Production rules, or "if ... then" rules, have proved to be a good method for knowledge based control within a module. A typical rule is made up of an antecedent and a consequent. The antecedent consists on one or more tests of information in the knowledge base. If all the tests in the antecedent are true then the consequent is executed. The consequent consists of one or more actions which can affect the knowledge base or execute an image processing algorithm. For example, a preprocessing rule may be: if (high_freq_noise) then run (median_filter). An example of a segmentation rule may be: if (contrast = low) then run (texture_segmentor). Production rules makes segmentation knowledge driven, however, it does provide increased flexibility and expandibility since production rules can be added or deleted as need be.

A knowledge driven architecture makes efficient use of all available knowledge about an image. It also provides an approach for processing information across a multiple of sensory information. In this way information from different sensors can be supplied to the knowledge base to be later retrieved from other sensors.

Finally, knowledge based segmentation can be designed to make use of information across multiple frames of a scene. In many cases information supplied by one frame can be used to help processing of successive frames. Using this system architecture, rules can be adaptively constructed throughout the processing stages to provide a more reliable segmentation procedure.

CHAPTER IV
INTERMEDIATE LEVEL PROCESSING

Intermediate level processing attempts to create symbolic representations from image features for scene understanding. It is a two step process, where image features are first extracted from a scene then grouped into a semantic form suitable for high level processing.

Intermediate level processing is a processing step used to interpret three dimensional scenes, notably for robotic vision. However, many of the techniques that will be examined in this chapter have only been applied to two dimensional scenes. Specifically, scene classification relies on statistical methods designed to classify non-complex 2 D images. In addition, the systems which use classification procedures have typically been developed for a narrow range of applications. They tend to be in essence, "dumb" systems. The major significance of the techniques used in scene classification, however, is that they lay a theoretical framework by which scenes that are more complex in nature can be processed and understood.

The classic distinction between pattern recognition and image understanding arises at the intermediate level of processing. For the pattern recognition community, segmentation procedures are essential for classification and ultimately scene understanding. However, the image understanding community, in an attempt to build general vision systems, generally agree that much richer representations of scene surfaces are needed. Intrinsic charcteristics of an image are an example. It is not that the image understanding community dismisses the utility of segmentation processes, but rather are attempting to create processes that are knowledge driven and amenable to symbolic processing. Regardless of the differences, the purpose of this chapter will be to review some of the important algorithms and processes used in solving the computer vision problem.

The following paragraphs will review some of the classic pattern recognition techniques used for image understanding. Most of ese tech-

niques have been developed for 2 D scenes. The emphasis will shift to analyzing some of the processes used to represent scenes in the three dimensional world that creates the 2 1/2 D sketch. Then a powerful computational algorithm will be reviewed that has application throughout all levels of image understanding processing. Finally, implementation using parallel architectures and optical processing will be referenced.

A. SCENE CLASSIFICATION

Much of the work in computer vision in the two-dimensional world, including document processing, radiology, industrial automation, remote sensing, tracking, etc., has developed because of the advances in pattern recognition techniques. These techniques have been characterized into two general forms: statistical (decision-theoretic) and syntactic (structural). Both of these methods attempt to match feature characteristics of a scene with 2 D mode based on some knowledge representation.

1. Statistical Pattern Recognition

The two general approaches in statistical pattern recognition are correlation and feature extraction. Correlation is the process of comparing an input image against a reference (template) image for all possible positions in the input field of view. Since a template match is rarely ever exact, because of noise, quantization effects, etc., a distance measure, $D(m,n)$, is commonly used. If we represent the image field as $F(j,k)$, and the template as $T(j,k)$, then the mean-square difference or error can be defined as,

$$D(m,n) = (F(j,k) - T(j-m,k-n))^2 \qquad (20)$$

A normalized cross correlation can be derived from Equation (20) that makes the correlation invariant on position such that, a template match is said to exist if $NC(m,n) \geq T(m,n)$, where $T(m,n)$ is a threshold level. One of the major limitations of template matching is that an enormous number of templates must be matched to account for changes in rotation and magnification of template objects.

Parallel implementation of simple template matching using correlation has been proposed on the GAPP system.[7] The correlation of a binary image is performed in a similar fashion as convolution except that bit-wide exclusive OR operations are used instead of multiplications.

The use of optical processing in pattern recognition/classification has become more of a reality over the past few years. Optical pattern recognition (OPR) has always offered the advantages of high speed and parallel processing with the basic linear systems operations of Fourier transformations and correlation being its hallmark. Research in recent years has considerably broadened this repertoire. Because of the introduction of sophisticated optical hardware and improved optical engineering techniques, many OPR prototype systems are now being developed.

The correlation method was one of the first techniques used for optical pattern recognition. The schematic in Figure 52 shows the process to obtain the correlation function in an optical processing environment. The input image f(x,y) is given at plane P1. The output at plane P3 is the Fourier transform of the product of the Fourier transform F(u,v) of the input function and the filter function H*(u,v) (complex conjugate of H) of plane P2. This can be expressed as,

$$s(x,y) = \mathcal{F}\{F(u,v)\ H^*(u,v)\} = f \circledast h \qquad (21)$$

where $\circledast$ means correlation. The output s(x,y) at plane P3 is found to be f(-x,-y) or an upside down and reversed image of the input function f(x,y). This results because an optical system can only perform forward not reverse, Fourier transforms. However, since h(x,y) is real and H*(u,v)= H(-u,-v) then the correlation is seen to be equivalent to a convolution without the coordinate reversal of one of the functions.

We can implement the correlation by holographically recording the pattern on a transparency at plane P2 and making it proportional to the reference object to be matched. This results in a matched spatial filter (MSF). The presence of a peak of light in the output plane at P3

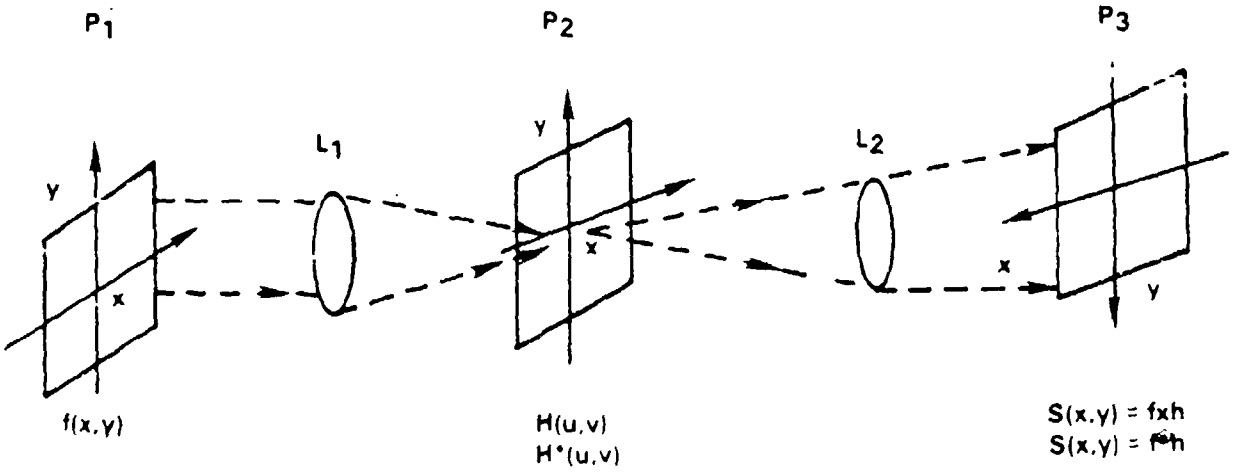


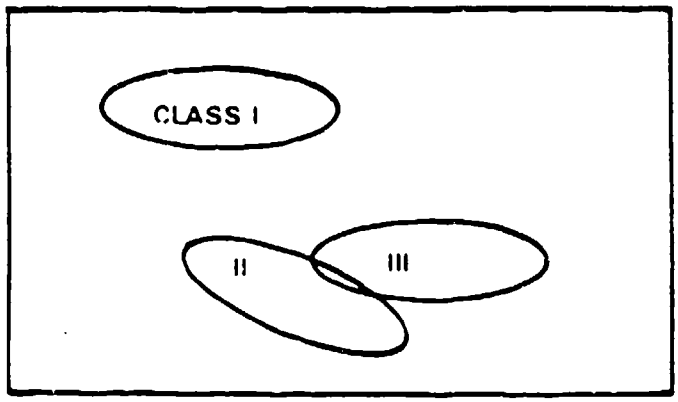
Figure 52. Schematic Diagram of an Optical Pattern Recognition Correlator

indicates that the reference object is present in the input image, and the location of the peak denotes where the reference object is in the input scene. Since the system in Figure 52 is linear and shift-invariant, superposition holds and thus the system is capable of recognizing multiple occurences of an input object.

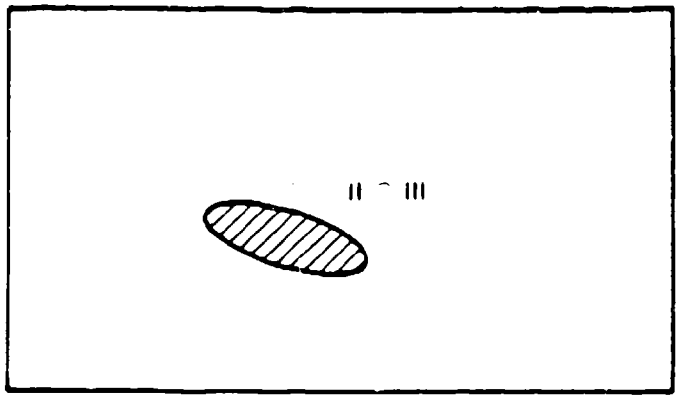
A matched filter is deterministic in nature. It does not include statistical features of the patterns to be classified. In particular, the matched filter is sometimes too sensitive to differences between patterns which are required to be grouped together as one class of objects. Many times it incorrectly matches patterns that look alike but are quite different. The problem is to find unique features so that the interclass variations of one statistically distributed object becomes a minimum, and the seperation of different classes a maximum. Refer to Figure 53 . For completeness, it should be mentioned that there has been much work in the area of matched filtering for stochastic image fields.

The second technique used in pattern recognition is feature extraction. Feature extraction can be described as a process of obtaining a feature vector, which consist of a set of scalar features, implementing a feature extractor and finally using a classification scheme. The features can include: geometrical features, Fourier coefficients, Mellin coefficients, moments, etc. The feature extractor attempts to extract the most prominent features, which usually involves projection of the measured feature vector onto one or several discriminant vectors by a vector inner product operator. Finally, the classifier then determines the object class from the projection value(s) obtained. Some typical classifiers are: minimum distance, nearest neighbor, minimum probability of error, etc. The effectiveness of these methods usually depend on the type of imagery, noise and clutter, target-type, speed requirements, hardware limitations, etc.

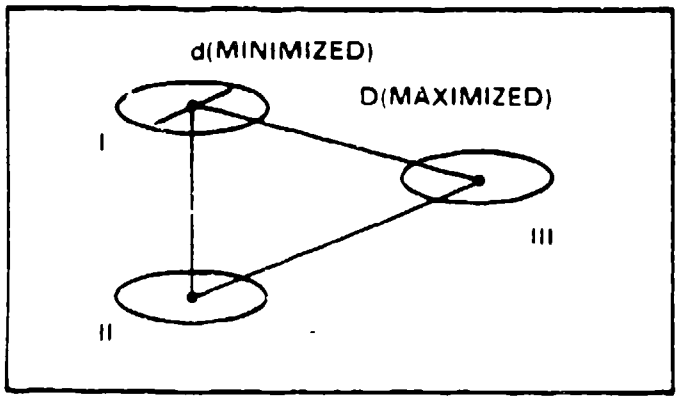
Two techniques mentioned for constructing the feature vector were moments and Fourier boundary descriptors. The geometric moments of an input object $f(x,y)$ are defined by,



a) FEATURE SPACES W/OVERLAPPING CLASSES



b) REGION WHERE MORE SOPHISTICATED TECHNIQUES ARE NEEDED



c) FINAL CLASSIFICATION

Figure 53. Classification Problem in OPR

$$M_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p y^q f(x,y)\, dx\, dy \qquad (22)$$

and the feature vector for an object in class i is denoted by M_i (its elements are the M_{pq} in Equation 22). In a digital environment, Equation (22) would be expressed as,

$$M_{pq} = \sum_x \sum_y x^p y^q f(x,y) \qquad (23)$$

where f(x,y) equals zero or unity for a binary image. The moments of a binary image are found by digitally summing over the image pixels weighted by the $(xPyQ)$ monomial masks. In addition, a set of moment equations can be derived that are invariant under translation, rotation, and magnification. These invariant moments can be used in the classification and matching process irrespective of the object's spatial location, rotation, and/or magnification.

It is possible to envision a parallel architecture to perform moment calculation. After the pixels that comprise the object boundary are known, then the moments of the image can be found by computing the moments in each processing element separately and then summing across the processing elements using recursive doubling.[33] This scheme requires that each PE know its absolute position in the configuration since the weighting of one of the moments in each PE is dependent upon the the PE address.

The moments can also be calculated optically on the system in Figure 54 . With different monomial masks $k(x,y) = (xPyQ)$ present on different spatial frequency carriers at P2, then the output pattern is simply,

$$\int f(x,y)\, k(x,y)\, dx\, dy \qquad (24)$$

which corresponds to the moments of the P1 input f(x,y), each located at a spatially different position in P3. Researchers[34] have developed ways to synthesize these masks.

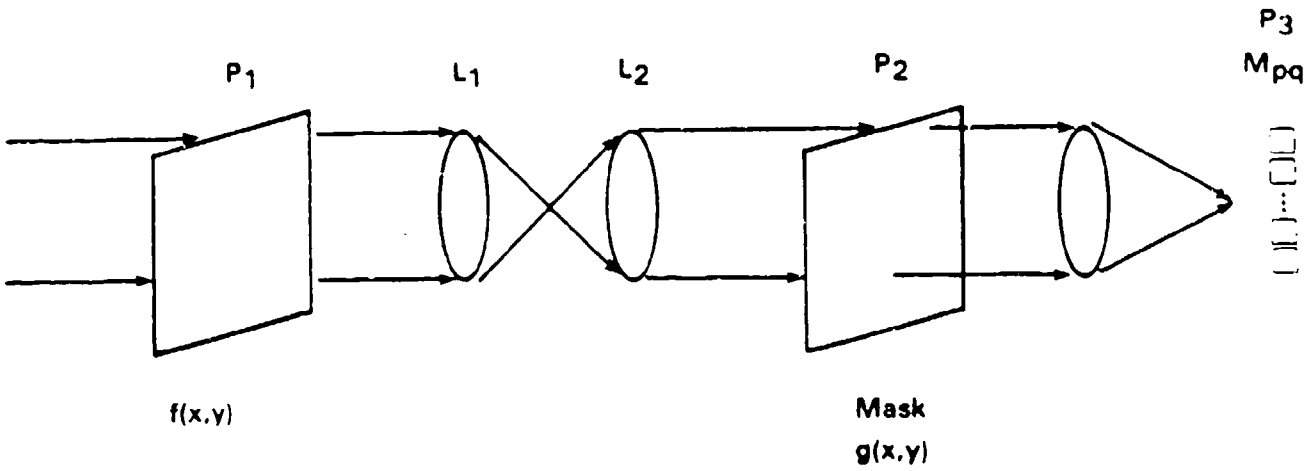


Figure 54 Optical System to Compute the Geometric Moment Failure

The moments calculated up to a particular order make up the feature vector. The feature extractor attempts to extract the most important features of a class. The feature extractor consists of transforming the observed moments, M' with the scaling vector, W^t, into a Fischer space by,

$$M = W^t M' \tag{25}$$

which reduces the number of features for easier discrimination. Estimates of class i of the input object are obtained from Fischer projections. A second level classifier is then used, i.e., the Mahalanobis distance, to minimize the distance between the input vector, M', and the reference vectors, M_j, for each class i.

The Fischer linear discriminant function has been used successfully in pattern classification. It is a linear function that provides the maximum ratio of between-class scatter to within-class scatter. For example, suppose that we want to classify a pattern into either class i or j, then it is possible, using Fischer's linear discriminant, to partition the feature space into two regions by a hyperplane decision surface. If this surface is positioned correctly, it provides a simple method for classifying patterns into one of two classes.

There has been research into implementing Fischer's linear discriminant function using a systolic array.[35] They have handled the classification problem for a more real-world, complex situation where there are more than two classes to discriminate.

Finally, a hybrid optical/digital architecture has been suggested to perform the classification procedure. Excellent performance, over 90% correct class recognition,[34] has been obtained with this parallel approach. Researchers are now in the process of prototyping this concept.

Fourier boundary descriptors have also been used as a feature vector in the classification process. The Fourier descriptors can be shown to be closely related to moments through the joint characteristic function. The Fourier descriptors are determined from the outline or con-

tour of the input image. Obviously the scene in the image would already have to be segmented in order to arrive at its contour.

To determine the Fourier descriptors, the contour line of some region or scene is described as a complex function of arch length t as depicted in Figure 55 so that,

$$z(t) = \mathrm{Re}\left[z(t)\right] + i\ \mathrm{Im}\left[z(t)\right] \tag{26}$$

This function is periodic with respect to its perimeter T and band limited because of its finite number of sampling points. Therefore it can be approximated by a Fourier series with (N+1) coefficients,

$$z(t) = \sum_{m=-N/2}^{N/2} C_m e^{im\omega t} \tag{27}$$

with $\omega = 2\pi/T$. The Cm are defined by,

$$C_m = \frac{1}{T}\int_0^T z(t)\, e^{-im\omega t} \tag{28}$$

As the values z(t) are complex numbers the positive and negative Fourier coefficients, Cm, are independent and can be represented as,

$$C_m = |C_m|\, e^{i\phi_m} \tag{29}$$

It can be shown that a set of Fourier descriptors can be derived from a truncated set of Fourier coefficients,

$$\tilde{C}_m = \frac{|C_m|}{|C_1|}\exp i\left[\phi_m + (1-m)\phi_2 - (1-m)\phi_1\right] \tag{30}$$

which are invariant with respect to location, size, and rotation.[36] The Fourier descriptors can be used as a set of features in the classification process.

The Fourier descriptor calculations are based on Fourier transform theory and consequently has the potential of being imple-

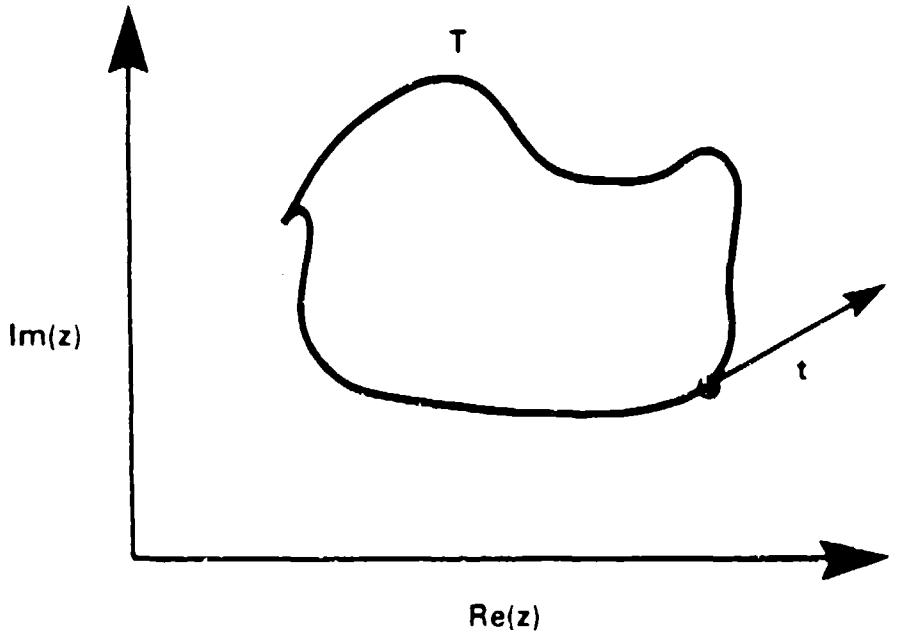


Figure 55. Contour Line as a Complex Function

mented in a parallel environment. The differences would reside in programming the PE's to calculate the invariant Fourier descriptors.

The calculation of the Fourier coefficients results in a large number of features since the number of Fourier coefficients is proportional to the space bandwidth product of the input image, which can typically approach 2.5×10^5. Usually one is forced to use a technique which reduces the number of features for easier classification. Researchers in optical processing have had success with the wedge ring detector (WRD) sampling system for reducing the number of features for the classification process. The details of the WRD can be found in.[34] Experiments have shown reductions from a bandwidth of 2.5×10^5 to 64 features. Then a Karhunen-Loeve transformation can be used to provide an efficient intra-class discrimination for feature extraction, while some type of discrimant function can be used for final classification.

Another process used for classification has been the Hough transform. Originally the Hough transform was used to detect lines[37] and later was used to detect curves that could be expressed in an analytic form, for example, circles, ellipses, etc. However, researchers[38] extended the application to detecting, segmenting and classifying arbitrary shapes in the analysis of real images. Given an arbitrary shape, S, the generalized Hough technique provides a mapping from the orientation of an edge-element to the set of instances of S, modified by location, rotation, and uniform scaling, which could have given rise to that edge element. This mapping allows all local evidence for a particular instance of S to contribute to global decisions about the figure. In practice, the Hough transform is similar to implementing a generalized matched filtering strategy, that is, template matching.

The algorithm for the generalized Hough transform can be generally described as constructing a parameter space from an image space, where the parameter space is used to locally describe the shape. The generalized Hough transform can also accomodate translation, rotation, and scale variations in images. A attractive feature of this transform is that it will work even when the boundary is disconnected due to noise o

occulsions. This is generally not true for strategies which track edge segments. Finally, the generalized Hough transform can be implemented in parallel.

The Hough algorithm for detecting and classifying arbitrary shapes, with fixed orientation and scale, can be described as follows. Given an arbitrary shape, depicted in Figure 56 , then for each point on the boundary with the gradient direction, ϕ, increment a point $\vec{a} = \vec{x}+\vec{r}$. Solving for $\vec{r}$ we get $\vec{r} = \vec{a}-\vec{x}$. The fact that $\vec{r}$ varies in an arbitrary way means that the generalized Hough transform for an arbitrary shape is best represented by a table called the R-table. The R-table is constructed by first choosing a reference point, $\vec{y}$, for the shape. Then for each boundary point, $\vec{x}$, compute $\phi(\vec{x})$, the gradient direction, and $\vec{r} = \vec{y}-\vec{x}$. Store $\vec{r}$ as a function of ϕ. The algorithm can be stated as:

(1) construct a R-table for the shape to be matched

(2) for each edge point (using an edge operator that supplies direction) in the image

  (a) compute $\Phi(\vec{x})$

  (b) calculate the "predicted" reference points

  (c) increment an accumulator array, $A(\vec{y}) = A(\vec{y}) + 1$

(3) maxima in the accumulator array indicates the translation

Rotation and scaling parameters can also be included by adding these two parameters to the shape description. This results in making the accumulator array increase to four dimensions corresponding to the parameters ($\vec{y}$,s,phi) and an increase in the information in the R-table. Many other improvements and variations can be added to the generalized Hough transform to consider it as a valuable computational tool for classification and intermediate level processing.

Finally, the Hough transform has the potential of being implemented in a massively parallel computing network. The PIPE system[39] has been used to perform the Hough transform. The idea is that all the "voting" (i.e. incrementing the accumulator array) can be hardwired to be performed in one step.

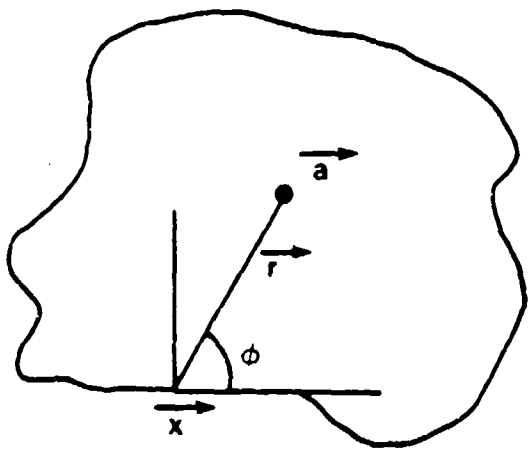


Figure 56. Geometry for the Generalized Hough Transform

Researchers[40] have also investigated using a neural network to solve this highly computational problem. The idea is to transform from a complex information flow of voting to a complex wiring architecture carrying simple excitatory and/or inhibitory responses.

Finally, there has been a proposal to implement a projection processor[41] to calculate the Hough transform in an optical environment.

2. Syntactic Pattern Recognition

In many situations, implementation of statistically based pattern recognition techniques for an image understanding system does have the tendency to become computationally intensive. As the image (pattern) to be analyzed becomes increasingly complex, the number of patterns, each with a n-dimensional feature vector, grows. The syntactic approach to pattern recognition attempts to solve this problem by representing a complex pattern by its simpler subpatterns (pattern primitives) and then apply the appropriate statistical methods to the subpatterns.

In syntactic methods, a pattern is represented by a sentence (a string or a tree) in a language specified by a grammar. The language, which is used to describe the structure of the patterns, is referred to as the pattern description language. The rules governing the composition of primitives into patterns are specified by a pattern grammar. After each primitive within the pattern is identified, the recognition process is accomplished by performing a syntax analysis, or parsing, of the "sentence" describing the given pattern to determine whether or not it is syntactically (or gramatically) correct with respect to the specified grammar. It is common to perform some preprocessing and segmentation operations on the raw image data before the parsing procedure is implemented.

The syntactic approach to pattern recognition provides a capability for describing a large set of complex patterns by using small sets of simple pattern primitives and grammatical rules. Also, one of the more attractive aspects is the recursive nature of a grammar. A grammar rule can be applied any number of times, so it is possible to express in a very compact way some basic structural characteristics of a large domain of sentences.

An alternative way to represent the structural information about a pattern is to use a relational graph. A relational graph is used to describe a scene in more detail, where relations between various subpatterns and primitives are specified. In a relational graph the nodes represent subpatterns and branches represent relations among subpatterns. A relational graph with such indications is called a directional graph.

The two key areas of importance in syntactic pattern recognition is the selection of: 1) subpatterns and 2) trees or relational graphs along with an efficient procedure for analyzing tree and graph structures. For line patterns or patterns described by boundaries or skeletons, line segments are often suggested for pattern primitives. For example, a line segment can be characterized by the locations of its beginning (tail) and end (head), by its length, and/or its slope. A curve segment can be described in terms of its length and curvature. Finally, shape and texture have been used as pattern primitives for describing regions.

After pattern primitives are selected, the next step in describing the pattern is the construction of a grammar. As mentioned previously, grammar is used to generate language. Unfortunately, as the descriptive power of a language is increased so is the complexity of the syntax analysis system (i.e. recognizer or acceptor).

Finally, in relation to practical applications, stochastic languages and error-correcting parsing have been suggested.[42] Stochastic languages are used to resolve the uncertainties that arise due to the presence of noise and variation in the pattern measurements, segmentation error, and primitive extraction error. Generally, stochastic languages entail associating probabilities with grammar rules. For example, if a sentence is found to be generated by two different pattern grammars, the ambiguity can be resolved by comparing the probabilities of the sentence in the two grammars. Then, a maximum-likelihood or Bayes decision rule will yield the final recognition.

Error-correcting parsing handles noisy and distorted patterns by using similarity measures. Similar to the template matching method in statistical pattern recognition, a similarity or distance measure can be

defined between a sentence representing an unknown pattern and a sentence describing a prototype pattern. Recognition of the unknown pattern can be carried out on the basis of the maximum similarity or minimum distance criterion.

A general procedure using primitives for shape description has been suggested. For example, if a texture pattern can be modeled as an arrangement of primitive patterns (deterministic or stochastic), then a syntactic method can be applied to texture modeling and discrimination. Gray level values, 1, of either single pixels or a homogeneous group of pixels (NxN) may serve well as the primitives. From a structural point of view, texture is the placement of structured subpatterns. To account for this characteristic, pictures are divided into fixed-size windows. Refer to Figure 57 . A grammar is then used to characterize the windowed pattern (KxK) of the given texture. Assuming that the window is of size (KxK), there are, $1k^2$, possible patterns. The set of all the windowed patterns of a particular texture is a subset of the patterns. A high dimensional regular grammar, for example, a tree grammar, is suitable for this characterization of textured patterns.

To create the tree representation, each pixel in a (KxK) window is made to correspond to a node. Hence, a pattern primitive becomes the assigned label to its corresponding node. A tree structure can be used to define the window and the labels on the nodes are used to define the pattern. The tree grammar can then be formulated that will generate the tree representation. Obviously, the choice of the tree structure determines the complexity as well as the effectiveness of the constructed tree grammar.

An example of a noise-free pattern is shown in Figure 58 . The pattern in (a) has the tree representation (b) for the tree structure model in (c). The tree grammar, G, used to generate the tree representation in Figure 58 is given in Figure 59 . V_1 is the set of terminal and nonterminal symbols, r is the rank associated with the symbols in V_1, P_1 is the set of production rules, A_1 is the starting symbol, and V_T is the set of terminal symb .

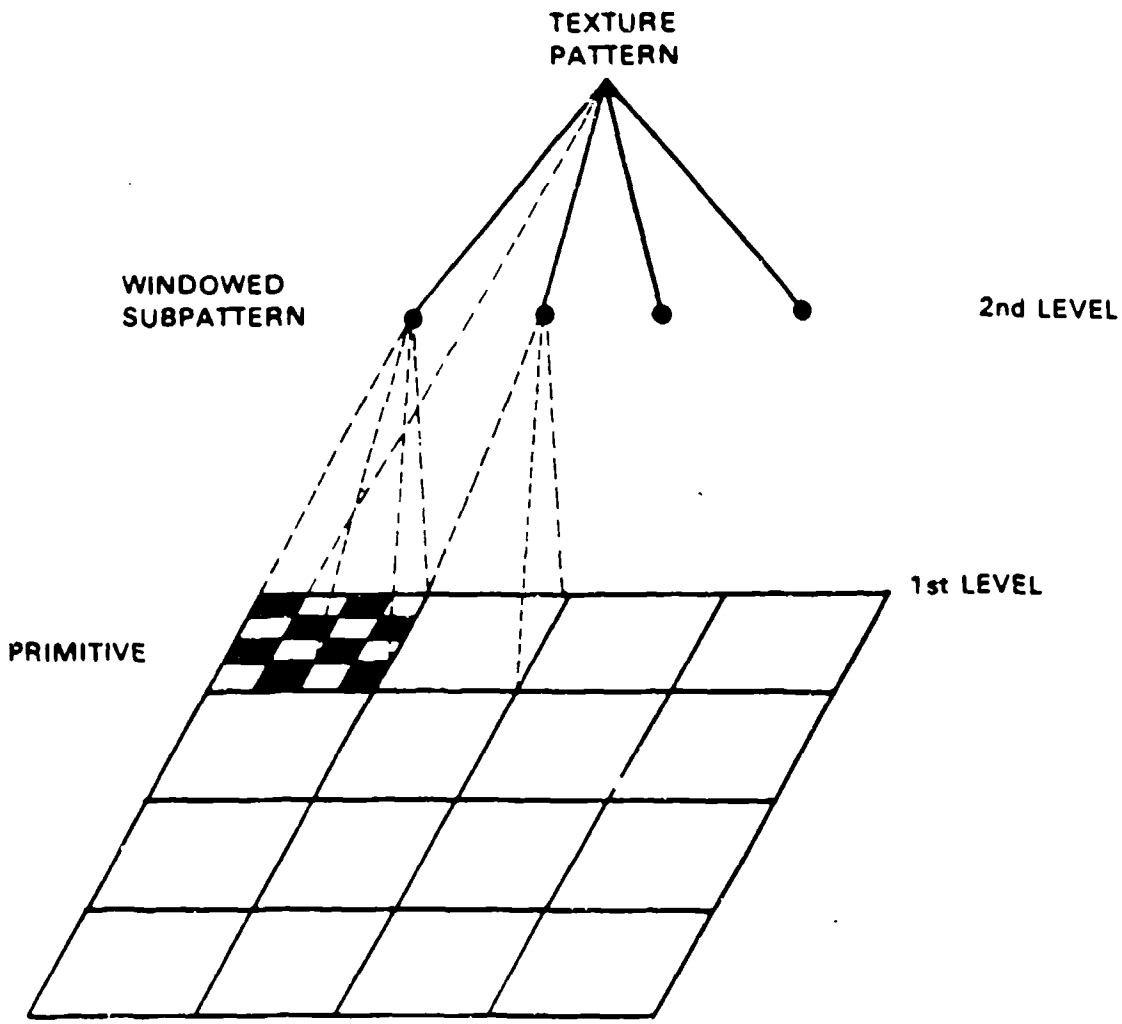


Figure 57. Structure of a Texture Pattern

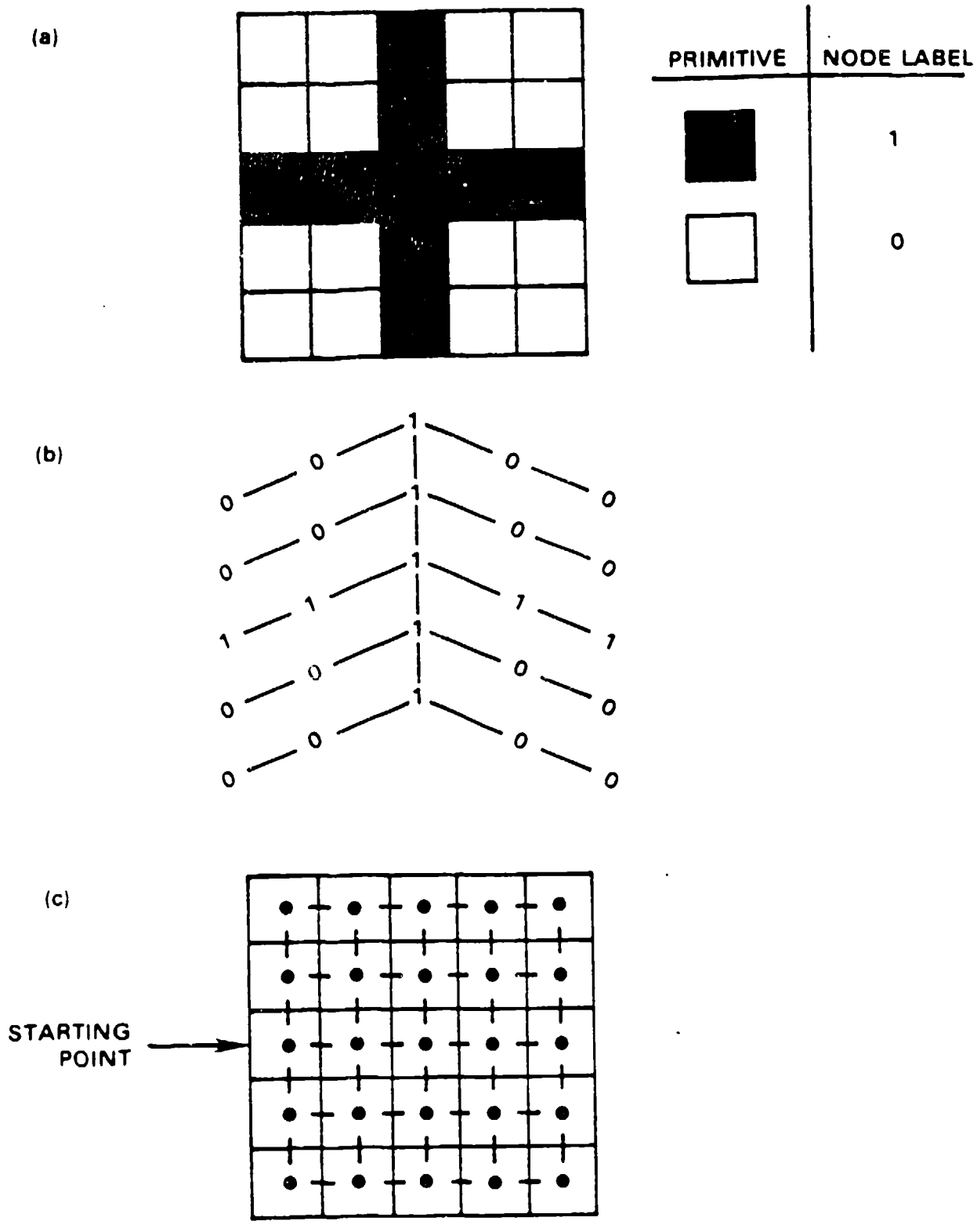


Figure 58. (a) Pattern, (b) Its Tree Representation, and (c) The Tree Structure Model

$$G = (V_1, r, P_1, A_1) \text{ OVER } \langle V_t, r \rangle$$

WHERE:

$$V_1 = \{A_1, A_2, A_3, A_4, A_5, A_6, N_0, V_0, 1, 0\}$$

$$V_T = \{ \underset{1}{\blacksquare}, \underset{0}{\square} \}$$

$$r = \{0, 1, 2, 3\}$$

AND P_1:

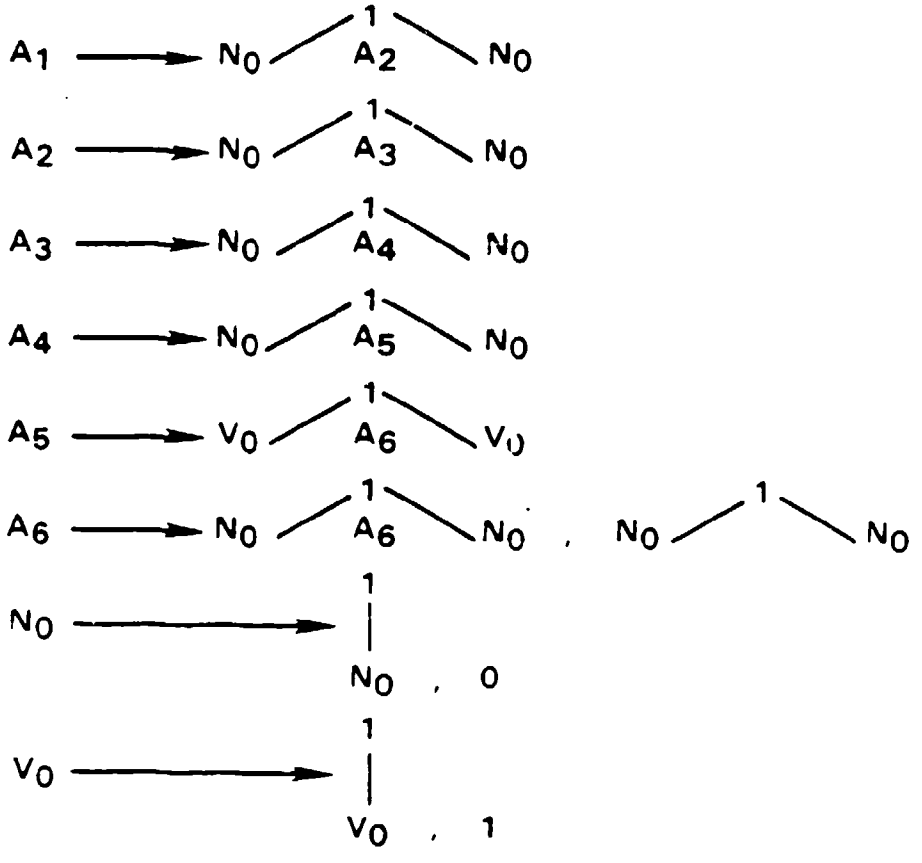


Figure 59. Tree Grammar for a Noise-Free Pattern in Figure 58

Since the parsing procedure for recognition is, in general, non-deterministic, it is regarded as computationally inefficient. Most of the early improvements have been in developing efficient sequential procedures. Of course, when a sequential procedure is used, the parsing procedure stops most of the time before a sentence is completely scanned which can prevent recovering complete structural information of the pattern. Another approach to reduce the parsing time is the use of parallel processing.

Researchers[43] have implemented Earley's parsing algorithm in parallel. Earley's algorithm is a non-backtracking tabular parsing procedure used for context-free languages. They were able to reformulate Earley's algorithm from a sequential to parallel form. Then they constructed an architecture to implement this parallel algorithm which operated in a pipelined and parallel fashion using VLSI technology. Earley's algorithm has had application in shape recognition.

B. THE 2 1/2 D SKETCH

The 2 1/2 D sketch attempts to represent complex three-dimensional objects in a scene. This is a classification procedure, where now three dimensional information is used to separate classes of objects within an image. Intrinsic surface characteristics such as depth, shading, orientation, and texture are properties used to recover regions of homogeneous groupings corresponding to three dimensional representations. The grouping process produces a symbolic form of the scene which later can be used for higher level processes, such as matching and scene interpretation.

The explicit representation of intrinsic surface characteristics marks a critical transition from pictorial information, created from the traditional 2-D pattern recogniton techniques, to information about the scene itself. Understanding the computational process by which intrinsic scene characteristics are recovered from an image has been a major focus of research by the artificial intelligence community interested in image

understanding for computer vision. The following paragraphs will discuss this in some detail.

It may be questioned on why I have chosen to incorporate intrinsic characteristics of the scene as part of intermediate level processing and not part of early level processing. The reason for this approach is that the intrinsic characteristics correspond to a 3 D world and therefore provide a richer description of a scene than a two-dimensional representation. In addition, these intrinsic characteristics have a propensity to segment and classify imagery all in one step because of the implicit information that they supply. Finally, processing on the intrinsic properties of an image is similar to the way a human gains knowledge about a scene. That is, many top-down, high-level assumptions, such as depth and orientation, are made when a human interprets a scene. These assumptions are based on previous learned experiences about the way the world is structured. The human visual system tends to utilize the intrinsic cues in the form of symbolic representations as an inference base for knowlege planning. Since the intrinsic characteristics, which produce the 2 1/2 D sketch, provide such a rich description of a scene they will be categorized as intermediate level processing.

The primary intrinsic images are surface reflectance, surface orientation, and incident illumination. Others include range, transparency, and specularity. The distance and orietation images correspond to a representation that was formally proposed by Marr,[1] called the 2 1/2 D sketch. The distance image gives the range along the line of sight from the center of projection to each visible point in the scene. The orientation image gives a vector representing the direction of the surface normal at every point. The reflectance image gives the albedo (i.e. the ratio of total relected to total incident illumination). Finally, the illuimination image gives the integrated incident illumination from all sources.

A simple example of intrinsic images and their usefulness in image understanding for a computer vision system can be described from experiments with a scanning laser rangefinder, which measures the properties of distance (depth) and apparent reflectance. Since range data is uncorrup-

ted by reflectance variations, and the amplitude data is unaffected by ambient lighting and shadows, extracting surfaces which are planar or have uniform reflectivity becomes an easier task. Such tasks are usually very difficult to perform reliably in gray level imagery, but with pure range and amplitude data, segmentation techniques, such as thresholding and region growing, perform well. Also, with the addition of some logic, relative to the information provided by the rangefinder, segmentation algorithms can be used to segment images into surfaces where reflectivity is constant and range is continuous. This is an important step for any robotic vision understanding system.

An image understanding system for machine vision requires the ability to interpret three-dimensional scenes from a two-dimensioanl representation. The grey level intensity values (shading) that represent a scene can be used to determine through dimensional surface shape. Simply stated, Horn[44] has devised a relationship between shape and intensity, I, at a point (x,y) in an image as,

$$I(x,y) = R(p,q) \tag{31}$$

where,

$$p = \partial f / \partial x \ , \ q = \partial f / \partial y \tag{32}$$

which he called the image irradiance equation. Refer to Figure 60 . This is a nonlinear first-order partial differential equation. The function R(p,q) includes information such as position of the viewer, distribution of light sources (assumed to be fixed), and the reflectance characteristics of the surface material. For a fixed distribution of light sources and fixed reflectance characteristics, the image irradiance equation associates a brightness value with each surface orientation. That is, a brightness value is assigned to each point in the pg- gradient space. The gradient refers to the orientation of a physical surface. Distance from the origin of the gradient space equals the slope of the

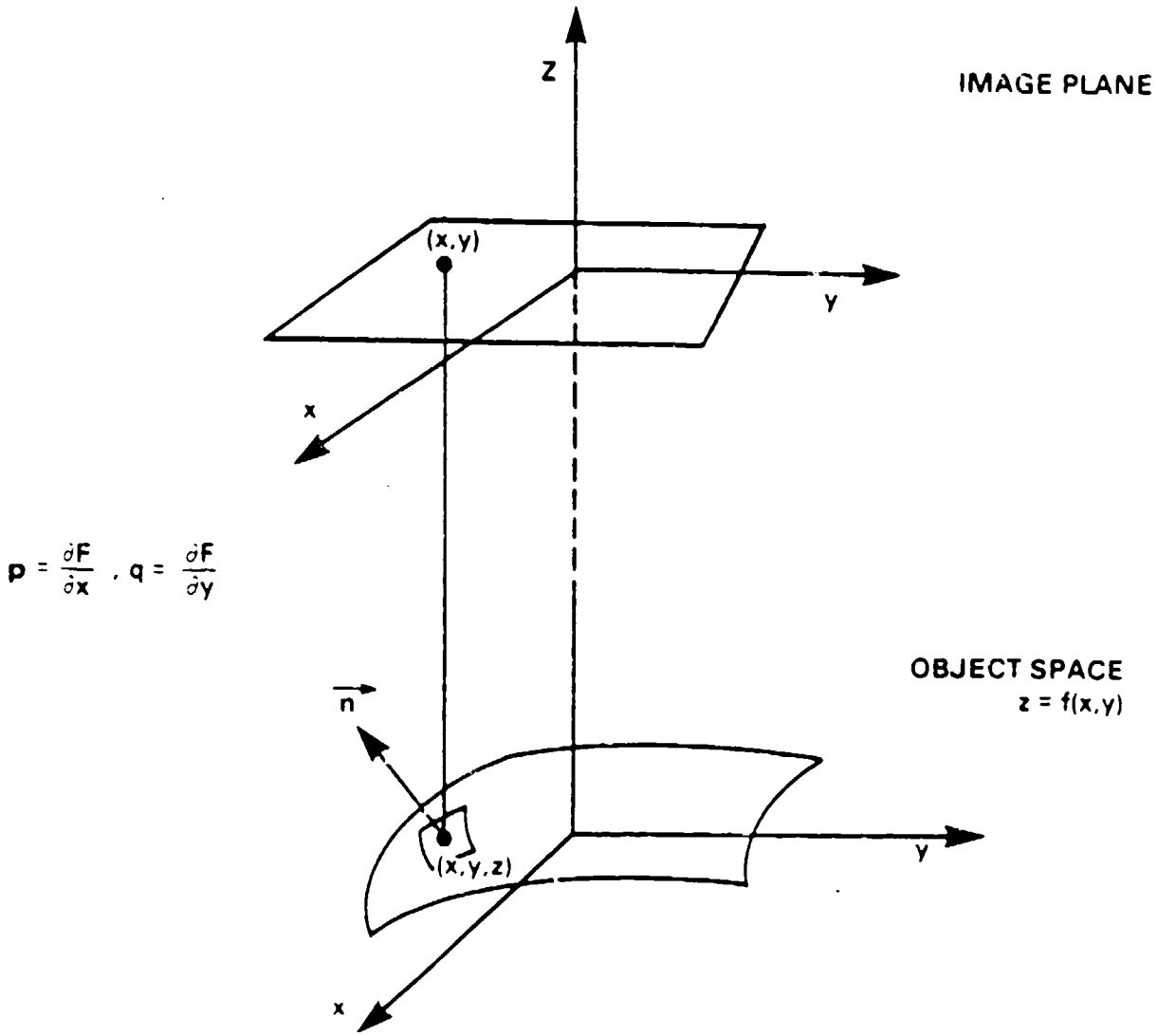


Figure 60. Viewing Geometry for Defining the Gradient Space

surface, while the direction is the direction of the steepest descent. This representation has been called the reflectance map.

Horn has applied the characteristic strip method for solving Equation (31) directly. The procedure to solve this nonlinear first-order partial differential equation is to set up an equivalent set of five ordinary differential equations, three for the coordinates and two for the components of the gradient. These five equations can be integrated numericallly along certain curved paths. The curve traced out on the object in this manner is called a characteristic. Then the process can be repeated for a series of steps in a specific direction. The shape of the surface is thus given as a sequence of coordinates on characteristics along its surface. Two disadvantages of this direct approach is that it is sensitive to noise and the base characteristic is data-dependent and thus it is difficult to determine the initial starting points.

An iterative technique has been developed by Ikeuchi and Horn[44] to solve this equation. Generally, they formulated the system into a variational problem with consistency (errors in gradients around a close loop) as a penalty term. Their method is less sensitive to noise and is also amenable to parallel implementation.

As suggested before, texture is a common property used to distinguish surfaces. Similar to the process by which the human visual system operates, texture elements of varying size in an image are commonly perceived as a three dimensional surface. The methods by which one determines surface orientation from this process is referred to as texture gradient techniques. Gibson[45] was one of the first investigators who used texture gradients in determining surface orientation. The general idea is that if the texture image is segmented into primitives, the maximum rate of change of the projected size of these primitives constrains the orientation of the plane. The direction of maximum rate of change of projected primitive size is the direction of the texture gradient. These ideas are related to the gradient space discussed before.

Witkin[46] has developed a computational approach to recover surface shape and orientation from the texture gradient. Witkin's approach was

novel in that he developed a procedure using texture gradients, similar to that proposed by Gibson,[45] and others;[47] however, he relaxed the constraints that defined the texture geometry so that his algorithm could be applied to a wide variety of natural scenes. In general, his technique recovers shape from texture contours in an image.

The general procedure that Witkin used to estimate the orientation of a surface was to develop a series of geometric and statistical models. A geometric model was used to provide a relation between the orientation of a texture primitive on a surface, the tangent to a curve on the surface, and the corresponding tangent in the image. Then he derived a statistical model that expressed the expected distribution of tangents on a surface and the corresponding distribution after projection. Finally, an estimator was derived that expressed the probability density function for surface orientation determined by the geometric and statistical models.

Witkin's first implementation was on planar surfaces, where the entire image surface orients itself uniformly. He used the D2G operator to obtain the texture contour points and then grouped the tangent directions into a histogram. An estimator was used to judge the maximum likelihood for surface orientation.

Witkin also applied his algorithm, with some modifications, to curved surfaces. However, rather than computing the tangent direction across the image, a local distribution was computed on a circular region surrounding each image point. Witkin had much success in both applications.

C. Relaxation

The purpose of a computer vision system is to produce a description of an image similar to that produced from a skilled observer. This requires application of many of the image processing and pattern recognition techniques already discussed. Some of them include image bandwidth compression, image restoration and enhancement, extraction of homogeneous regions, measuring features that characterize and/or classify these segments, and measuring relations between these segments (i.e. brighter

than, smaller than, etc.). The output of this complex sequence of events is far from being an array of gray level intensities that was initially captured by the imaging system. Rather, the output has been transformed into a representation that is more appropriate for high level processing), that is, a symbolic description such as a labeled graph or semantic network.

In addition to the information that is extracted from the scene, we also have supplied to us knowledge about the type of scene that we wish to describe (the model). This knowledge representation can also be transformed into a labeled graph. The solution then becomes one of matching the graph network of the scene to the graph network of the model. This global matching procedure is called relaxation, since it resembles an iterative procedure in numerical analysis. The use of relaxation is considered by many as a powerful algorithmic tool for symbolic image description and image understanding throughout all levels of processing. Some examples of where relaxation has been used is edge detection, shape from shading, optical flow, line labeling, semantic net matching, etc.

There are two classes of relaxation processes, the discrete and the continous (or probabalistic). The discrete simply associates a set (i.e. 0/1) of possible labels, or names, to an image part. The continous case involves assigning a likelihood (probability) to this association. Most of the relaxation techniques incorporated into modern vision systems use the continous case, especially when the image is complex in nature. Both of these classes utilize two general models to generate the labeling scheme. The first is the neighborhood model which determines which parts in the image directly communicate to other parts. The second is an interaction model which defines how an image part changes its labeling based on the labeling of its neighbors.

1. Matching Details

The goal of relaxation is to match elements in the image to elements of a model of that scene. Elements in the image can be representative of such things as regions, segments, edges, etc., that have been produced from earlier processing. Elements of a model of a scene are

constructed from knowledge about the type of imagery being processed. Knowledge describes how elements in an image are related. These descriptions (or rules) represent the model of the elements in a scene.

Let the model elements be denoted by a unit (U_j). The interconnection between these units may be used to give a fuller description of these elements. The model elements and their interconnection can then be considered as a graph. Likewise a graph can be constructed for the elements of the image (the names (N_j)). Relaxation attempts to match the model graph, $M(u_j)$, to the image graph, $I(n_j)$.

The links between the elements (nodes) indicate the relations between the nodes, i.e. above, below, larger than, smaller than. In addition, for every node, u_j, there is a corresponding probability vector assigning a measure of likelihood to the relation,

$$\vec{P}_i = \left\{ \vec{P}_i(m_1), \vec{P}_i(m_2), \ldots, \vec{P}_i(m_N) \right\} \qquad (33)$$

where N is the number of names in the image graph. The set of all vectors $\vec{P}_j$ is called the stochastic labeling of the set of units U.

A formula can be defined to rate how well elememts are matched in each graph. The criteria can be based on a variety of methods, some of the standard approaches use a simple weighted difference measure. The range of the match formula is $[0,1]$ so that a perfect match assumes a value of unity and no match obtains a value of zero. Let us denote this matching formula[48] as,

$$r(u,m) = \frac{c}{D(u,m) + c} \qquad (34)$$

with c being a constant equal to unity, and $D(u,n)$ being the difference measure summed over all features, defined as,

$$D(u,m) = \sum_{k=1}^{f} |V_{u_k} - V_{m_k}| w_k s_k \qquad (35)$$

for f features, w_k is the weighting, s_k is the assigned strength, and V is the k^{th} feature value.

The matching formula is used for two distinct purposes. First, it is used to obtain the initial probabilities, (i.e. the initial likelihoods of particular assignments),

$$\vec{p}_i^{(o)} \tag{36}$$

Second, the matching formula is used in an adaptive mode to measure the quality of a match. That is, the matching formula is needed to adjust for the compatability of the matching elements. Since the interaction between elements u_i and n_j is based on their connection (relation) to other elements, then the compatability function will be based on these relations (the links between the node elements).

The compatability function between every unit u_i with each name n_k can generally be given as,

$$\vec{q}_i \tag{37}$$

Actually $\vec{q}_i$ can be thought as representing what the neighbors of unit u_i "think" about how it should be labeled, whereas, $\vec{p}_i$ represents what unit u_i itself "thinks" about its own labeling. The difference between $\vec{q}_i$ and $\vec{p}_i$ represents an inconsistency and ambiguity that can be defined as the dot product between these two vectors,

$$J_i = \vec{p}_i \cdot \vec{q}_i \tag{38}$$

The goal, then, is to minimize this inconsistency function J_i (i.e., $\vec{p}_i = \vec{q}_i$ and $\vec{p}_i$ = unit vector). A global measure can then be defined over the whole set of units by averaging the local measures. By using the arithmetic average,

$$J = \sum_{\text{all } i,j} J_i \tag{39}$$

Finally an updating procedure is needed that will provide the best estimate of $\vec{p}_i$. In order to achieve this locally, the global inconsistency function can be minimized by attaching to every unit u_i a local gradient vector,

$$\vec{g}_i = \frac{\partial J}{\partial \vec{p}_i} \tag{40}$$

where the coordinates of $\vec{g}_i$ are functions of the vectors attached to the units u_j in the set $M(u_i)$. The iterative scheme, used in numerical analysis, to minimize the global inconsistency function can be described as,

$$p_i^{(n+1)} = p_i^{(n)} + \rho_n H^{(n)} \tag{41}$$

where ρ_n is a positive scale factor, and $H^{(n)}$ is the negative of the gradient given in Equation 40.

The procedure to implement the relaxation algorithm is shown in Figure 61 . The first step is to choose elements that are to be matched. Next determine the initial probabilities using all available features and relations with assigned model units using the matching function. Then begin relaxation to obtain the "best" $\vec{p}_i$, comparing it to a threshold, t. If $\vec{p}_i \geq t$ then make assignment (match) otherwise relax again.

Other researchers have used relaxation techniques,[17,49] successfully. The major differences between them are: (1) they utilize the same compatability function but a different updating function; (2) some define the compatability function differently while using the same updating function; and (3) some do not optimize J, so the updating function, while being similar, is not as "smart."

2. Relaxation and Parallelism

Relaxation is a labeling process which requires alot of computation since the labeling needs to be performed on a large amount of data. Similar to the way biological vision systems operate, a promising approach is to make the processes parallel. This means that each part of the image is to be analyzed and labeled independently of the others. One obvious

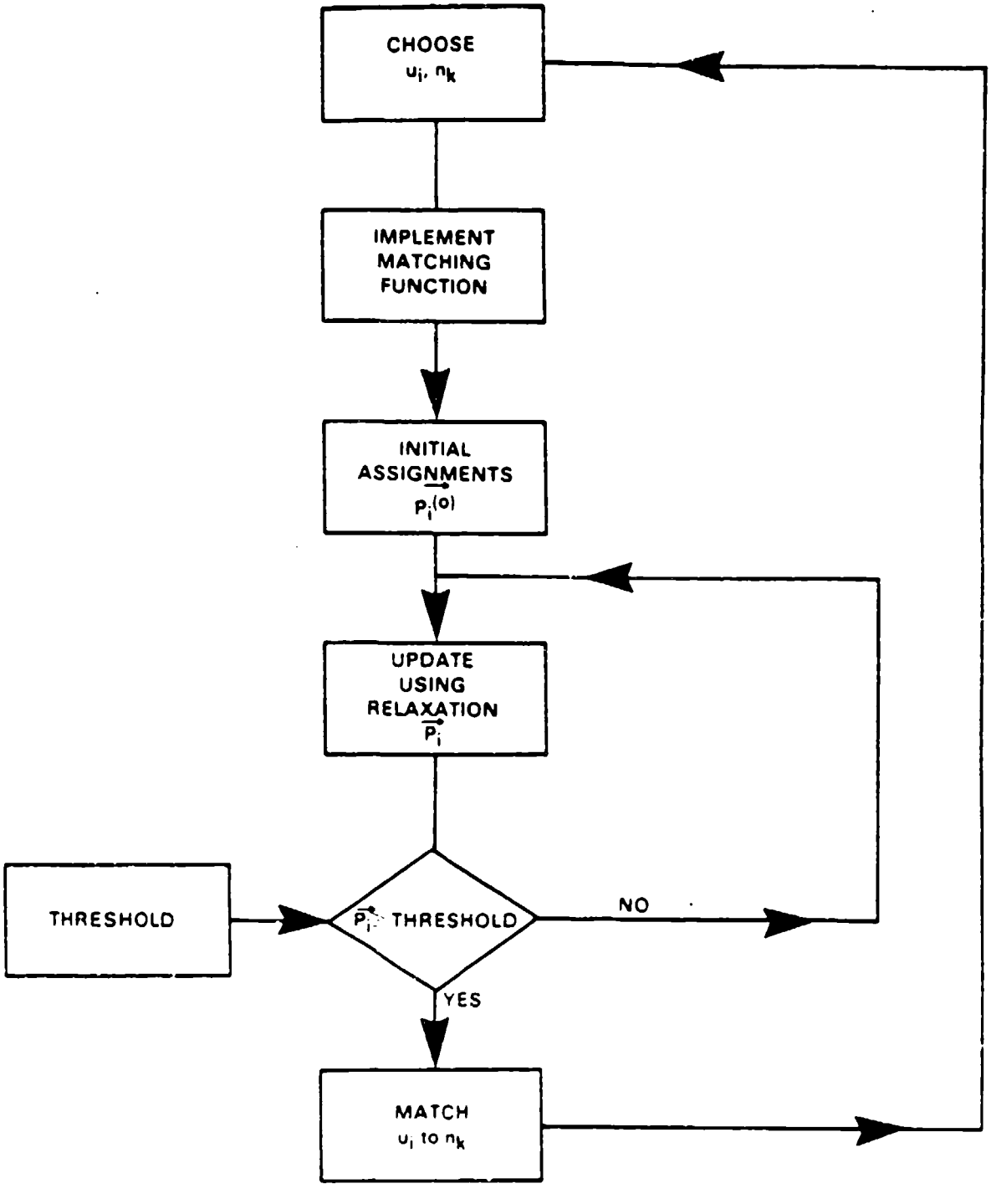


Figure 61. Symbolic Matching by Relaxation

shortcoming of this approach is that the full contextural information of the image is not fully utilized, which can lead to propogating errors in the image understanding process.

A solution to this problem, which relaxation addresses, is to assess the labelling probabilities for every part independently and then compare each parts' assestment to those of the other related parts, in order to detect and correct potential inconsistencies. Since both the assesment and the comparison can be done independently, we are still able to perform parallel operations. To maintain computional cost as low as possible, the comparisons should be local, i.e. among neighboring pixels, while maintaining information flow by iterating the comparison process. As previously mentioned, relaxation can be performed in parallel by using Equation 41.

Researchers[50] are currently developing an architecture to perform this higher level vision processing algorithm. It is called the semantic network array processor (SNAP). They are currently trying to implement discrete relaxation.

The SNAP architecture consists of an array of identical cells each containing a content addressable memory, microprogram control, and a communication unit. Figure 62 shows the SNAP architecture. SNAP's functionality is a product of two characteristics: associative processing and cellular array processing. Each cell contains memory control logic and communication logic. The cellular array is operated by the array controller, which in addition provides an interface between SNAP and a host computer. The cells can be microprogrammed so they can operate independently. The cells communicate via local buses. A cell's address is specified by its row number followed by its column number. Information in a particular cell can be retrieved either by its content (as in associative memories) or by the cell's address.

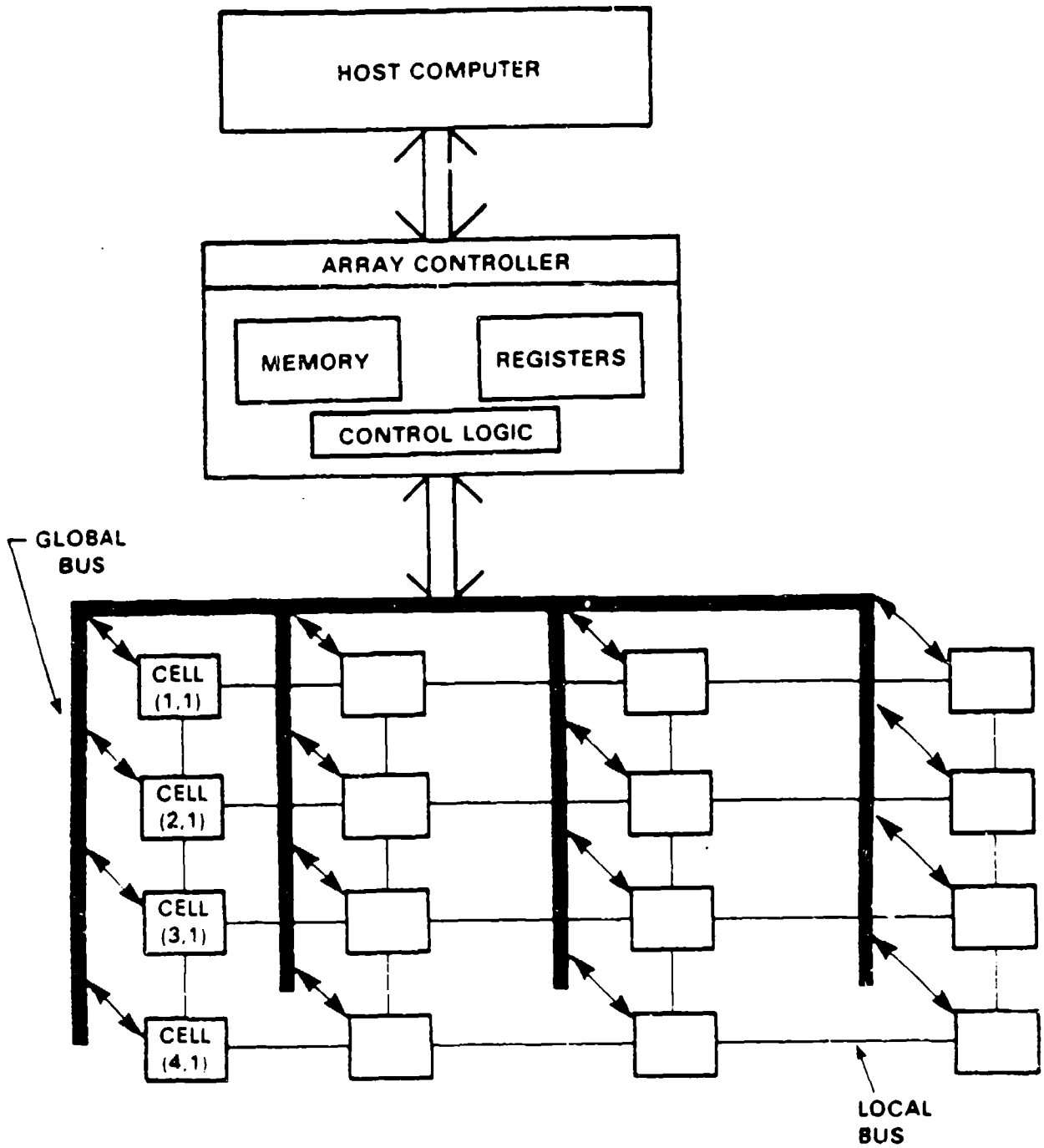


Figure 62. Architecture for SNAP

CHAPTER V

HIGH LEVEL SYMBOLIC PROCESSING: ATTEMPTS TO CREATE THE 3D SKETCH

Scene classification and interpretation are key processes in image understanding. For two dimensional scenes this is generally regarded as a matching process between features derived from the image and 2 D models. To be more specific, it is not a simple match, but rather it is a process of verifying that the model can give rise to a particular representation. Many of the computational techniques that have been described to accomplish this task have given excellent performance.

However, for more complex scenes, classification and interpretation is much more difficult. Information regarding intrinsic characteristics of the scene, as described by the 2 1/2 D sketch, provide valuable information to the image understanding process. Robot vision is an example, where the scene to be described is fundamentally three dimensional, involving substantial surface relief and object occulsion. As mentioned earlier, even two dimensional scenes that are complex in nature are problematic in terms of classification and interpretation.

The objective of high level processing for image understanding is to operate on the intermediate representations (i.e., the full primal sketch, 2 1/2 D sketch) derived from earlier vision processes for scene recognition, classification, and interpretation. The operations performed in high level processing attempt to group earlier processes into a form that is amenable to symbolic processing. Once the scene is restructured into symbolic form, the final step is to give the system "knowledge." The approach taken has been to "program" into the machine vision system a reasoning capability using artificial intelligence techniques.

Symbolic representations of image elements provides the easiest way for a computer to understand a complex scene. Symbols supply information such as form, structure, and groupings whereas processing over symbols supplies information such as relations, occurrences, and matching. Symbolic processing over image elements allows the vision system to make inferences about a scene, similar to the way a human would.

The human visual system is able to accept information from a multiple of sources, fuse the information into a pool of knowledge, reason across the knowledge, predict which of the sources will provide the most beneficial amount of new information, and direct resources to those sources to process new information. Often this information is the result of partial scanning, resource-limited processing, heuristic beliefs, and probabilistic assumptions. Thus, human processing uses information which is incomplete, uncertain, even incorrect. Each piece of information thus has some amount of belief associated with it. In other words, the belief is based on evidence. Making an inference about the world based on beliefs requires not only reasoning over the information but reasoning about the belief of the information and the evidence that the belief is based on.

The goal of an intelligent machine vision system is to manage the information that it gathers, similar to the way a human would. The vision system must be able to fuse, evidentially reason over, and control the processing of a variety of information that may be uncertain, incomplete, and even incorrect.

The two elements that are essential for high level processing are: (1) knowledge representation, and (2) association/conversion between knowledge representations and scene domains. Both of these processes involve some form of contextural processing. The purpose of knowledge representation is to store facts, relationships, and strategies for deriving them. A variety of knowledge representation schemes exist. Some of them include: (a) semantic networks; (b) object-attribute-value triplets; (c) rules; (d) frames; (e) logical expressions; (f) 2-D and 3-D structural representations; and (g) iconic. For example, a semantic network represents objects and relations between objects as a graph structure of nodes and labeled arcs. The arcs usually represent relations between nodes. Semantic nets are especially attractive as analogical representations of spatial states of affairs.

The association/conversion between knowledge reprsentations and scene domains is basically a matching process. The matching is conducted

between a model of the scene, derived from prior knowledge about the type of imagery, and the represenations derived from earlier vision processes.

A conceptual functional architecture for high level processing for image understanding is given in Figure 63 . The architecture consists of the following five modules: (1) feature organizer; (2) predictor; (3) planner; (4) inferencer; and (5) matcher. Each of these elements will be discussed in the following paragraphs.

The feature organizer derives relevant groupings and structures from an image without prior knowledge of its contents. The predominant processes that are used to perform these tasks are segmentation and feature extraction. The type of process depends on a number of factors, including data type, computational complexity, etc. The feature organizer performs the following functions. The first is that it partitions an image into sets of related features, which reduces the search space required for model selection and matching. Second, the relations formed by these processes can serve as reliable indicies to access both model and world knowledge bases.

The predictor applies expectations concerning what objects are expected to be present in an image and the way in which they might manifest themselves. The predictor aids in the matching process.

The planner generates a sequence of actions intended to facilitate the image understanding process. Planning is the process of ordering the operations such that their actions and resource needs do not conflict. The planner aids in making decisions as to what sources of information should be concentrated on, what processes to run, where and when to run them, what parameters to use, and which data to pass along.

Several planning strategies are applicable to high level processing for image understanding. There are hierarchial planning strategies, which involves a detailed description of the planning steps as well as a high-level conceptual specification so that there is a hierarchy of representations of a plan. The hierarchial planning procedure operates from the abstract to more specific so that the plan development is not initially computationally overwhelming. Given a high level goal with

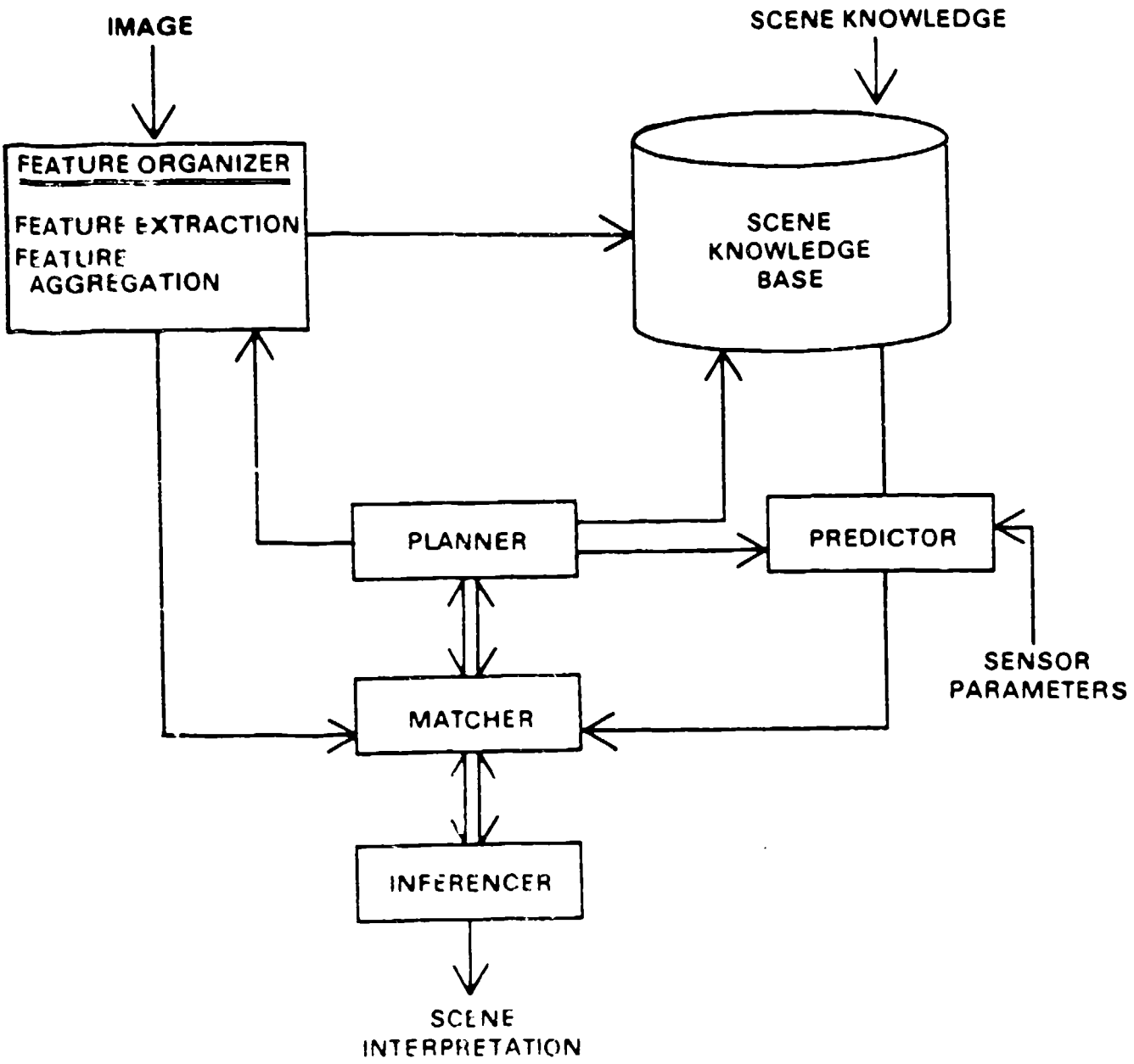


Figure 63. A High Level Processing Schema

constraints, the goal is expanded into a partially ordered net of subgoals and actions which will achieve the original goal. Then the subgoals are examined and compared to the internal world model to see if they are true. If they are not, the net is examined to determine if reconstructing the plan will make the subgoal true, and if not, the subgoal is made true by expanding it into more subgoals. This process continues until all goals are true. Intermitently the plan-net is examined for conflicts between goals (i.e. attempting to use the same resources at the same time). Often conflicts are resolved by reordering the plan-net. Finally, goals are kept parallel until a necessary ordering can be determined, and variables are not arbitrarily bound but have constraints placed on them until a correct value is found.

There are non-hierarchial planning strategies which generates only one representation of a plan. This technique does not distinguish between problem solving actions and other actions, and may employ goal reduction to simplify the problem. The non-hierarchial system may also use means-ends analysis to reduce the differences between the current state of the world and the result of executing the plan. Non-hierarchial planning strategies are usually implemented on simpler vision systems or systems which can be strictly defined.

The matcher forms relations between internal representations of visual information and world knowledge for the purpose of scene recognition and interpretation. Matching can occur between numerous representations of an image including iconic, geometric, and relational structures. Matching at the iconic level is carried out by template matching, as discussed in a previous section. Matching at the geometric level involves creating a correspondence between data and a parametized model. This matching is a process which determines which parameters of the model are best utilized for a meaningful representation of the data. An example of this type of matching is the feature extraction process used in statistical pattern recognition. A third form of matching involves relational structures. Since relational structures are often represented as graphs or networks, the matching problem can become one of graph or subgraph iso-

morphism. Some techniques include enumerative search of the tree of possible matches between nodes, and parallel-iterative refinement applied during search. An example of this type of matching is relaxation. Also, generalized structure matching may also be used where isomorphism is not possible due to missing or uncertain information. An example of this type of matching process was demonstrated in the syntactic pattern recognition process.

Control mechanisms are incorporated to ensure that the inference process proceeds in a logical fashion. Several techniques for control are relevant to high level processing. They are: (1) data driven vs. goal-driven; (2) hierarchial vs. heterarchical; and (3) parallel vs. serial.

Finally, the inference module is used to deduce the presence or absence of features within an image and for the ultimate recognition and interpretation of objects and scenes. A number of inference mechanisms are applicable to high level processing for image understanding. Predicate logic is a system for expressing propositions and deriving consequences of facts. Production systems are " if ... then ..." or situation-action pairs which are more flexible than first order predicate logic. Labeling schemes are another form of the inference process that is used to assign labels to images in a probability-like fashion. We have seen labeling schemes when we discussed relaxation. Finally, active knowledge used procedures as the elementary units of knowledge.

The five modules that have been described represent the functional approach that is being used for high level processing. Whereas in low level processing the source of data was scene primitives and features and in the intermediate level the processes involved grouping these primitives and features into symbolic form, high level processing involves operating over these symbolic representations. Generally, these symbolic representations are in the form of a semantic expression. Consequently, the type of operations performed involve contextural processing. The languages that are being used to perform this type of processing are LISP and PROLOG. These languages are well suited for symbolic data processing.

One of the important considerations for researchers working on prototype vision systems has been the design of a robust matching module. LISP has been used successfully as a pattern matcher. Pattern matching is the process of comparing symbolic expressions to see if one is similar to another. Although LISP has no pattern matching built in, LISP makes it is easy to write pattern matching procedures. The general strategy to accomplish this is to match a pattern to a model (datum) using some of the symbol-manipulating functions and matching procedures.

There are other facilities of LISP that make it attractive in designing the reasoning modules used in high level processing. Production systems or "if..then.." rules can be efficiently constructed in a LISP environment to provide an inferencing capability. Production systems are popular because they offer modularity and incremental growth characteristics.

A production system has three general components: (1) a data base; (2) a set of rules; and (3) an interpreter for the rules. Of particular importance is the matching of rules or patterns to the data base. Procedures in LISP for pattern-directed matching are called demons. Demons are procedures that are activated automatically when a value is placed or removed and/or modified from a database. Demons constructs have been an important development in designing a robust vision system.

CHAPTER VI
AN EXAMPLE

It would be very instructive to provide an example of the steps used in an image understanding process. This example is not meant to represent the most efficient processes nor the approach that would be taken in a real-time application but rather is intended to be pedantic.

Figure 64 (a) is an infrared image of a tank. Our purpose is to separate the tank from the background and to classify which type of tank it is. We can apply a segmentation algorithm that is sensitive to a particular characteristic of this type of imagery. The result of the segmentation algorithm would be a blob representation whose shape would grossly resemble the shape of the tank. However, information is lost as to the particular arrangement of parts of the tank, for example, the wheels, tread cover, turret. This is a common problem in an image understanding system.

In order to produce a reliable segmentation procedure it would be beneficial to use cues about the way image features are structured in the world. These cues are in the form of knowledge rules which reflect the way in which humans group objects. These rules are based on Gestalt psychology. For example, characteristics such as proximity, similarity, collinearity, and containment can be used as reasoning cues to aid in the image understanding process. This will enable easier classification and interpretation.

Figure 64 (b) is a primal sketch representation of the tank which is used to show the primitive characteristics of the image. As can be readily observed, additional processing is needed to interpret this scene. Figure 64 (c) is the result of further processing where the tank is segmented from the background but there is confusion as to the arrangement of tank regions. For example, the segmentation algorithm produced three separate regions for the tread cover. Obviously, there is a need to cluster these regions into a homogeneous grouping. Once this is accomplished it will be easier to proceed to the next step in the image understanding process.

Grouping the regions into a symbolic form is considered an intermediate level processing step. This representation will make it easier to perform high level processing for scene understanding. Figure 64 (d) shows an arrangement of regions that have been arbitrarily assigned, which has resulted from the segmentation algorithm. Applying the grouping rules mentioned earlier in a well defined order produces the graphic in Figures 64 (e) and 64 (f). Figure 64 (g) is the result of the final clustering procedure where the tank is segmented into three separate groupings relating to the three distinct regions of a tank - the wheels, tread cover, and turret.

Figure 64 (h) is a tree-like structure which represents the steps taken in region clustering. This symbolic representation is in a form of a semantic net which aids in high level processing. The primal node groups all the clusters of nodes into a single node which represents the entire image. The other nodes are clustered according to the rules mentioned earlier. Each node is a symbolic representation of a region and its attributes. Records are kept as to the relation of one node to the next. Evidence is accrued relating to these records for merging purposes. Finally, matching this tree-like structure to a model results in classification of the tank.

Figure 65 summarizes the steps for scene understanding in the tank example. Of particular importance is where knowledge is incorporated into the processing steps. For this example, which can be indicative of other cases, knowledge has been implemented at segmentation. In some cases it should be implemented as early as preprocessing. In addition, since the image understanding process is goal-driven as well as data-driven, knowledge that is incorporated into semantic net for high level processing can also be returned to other lower level processing for planning purposes.

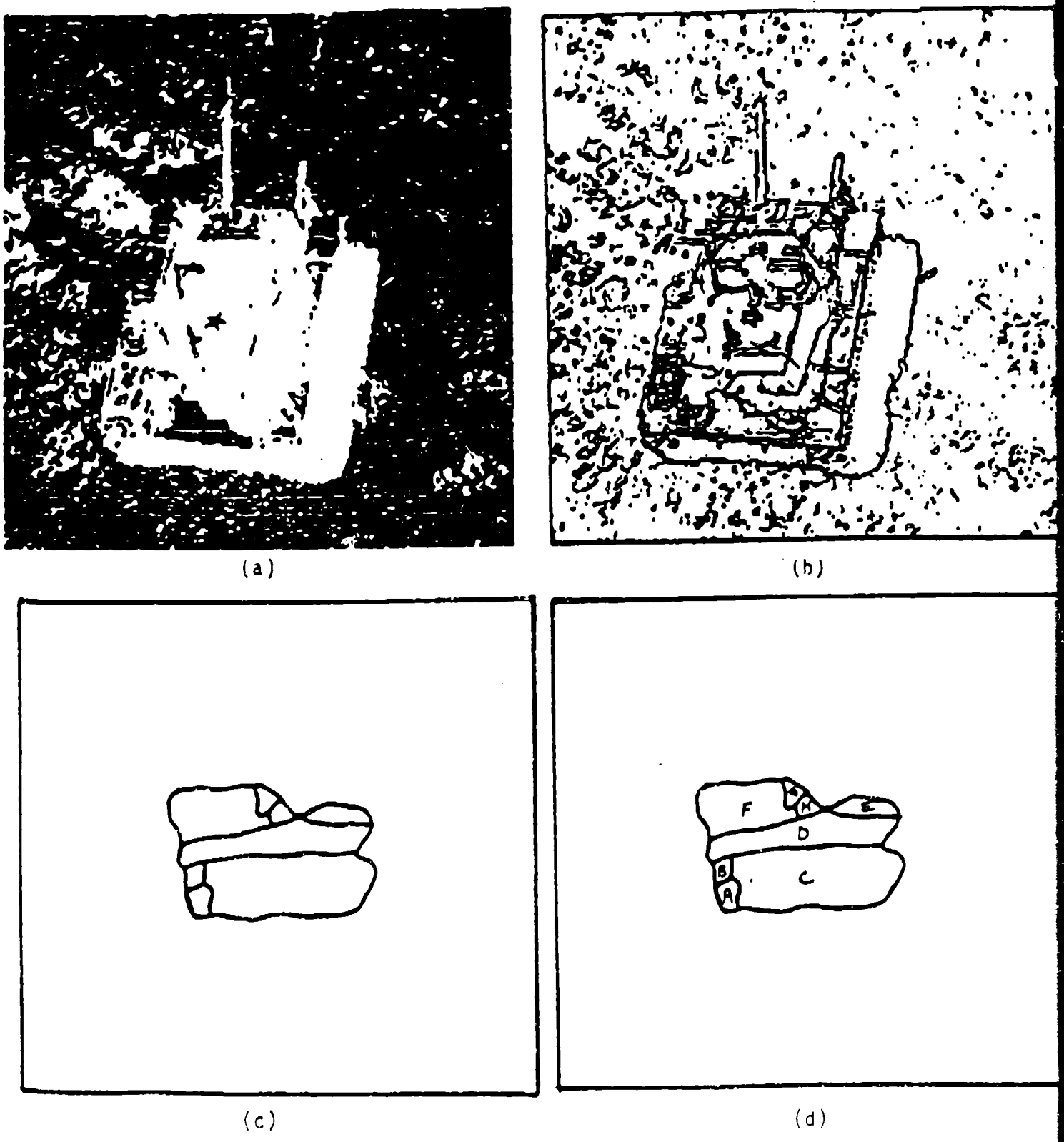
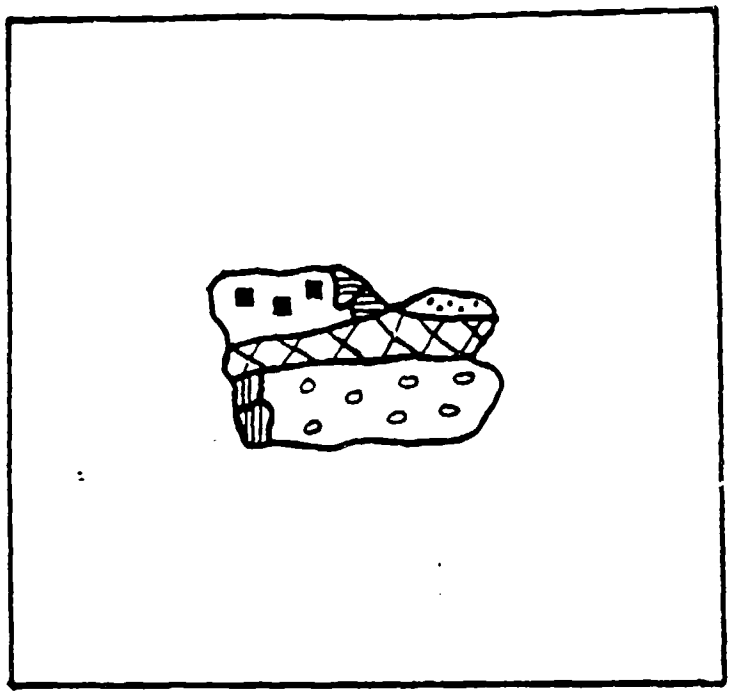
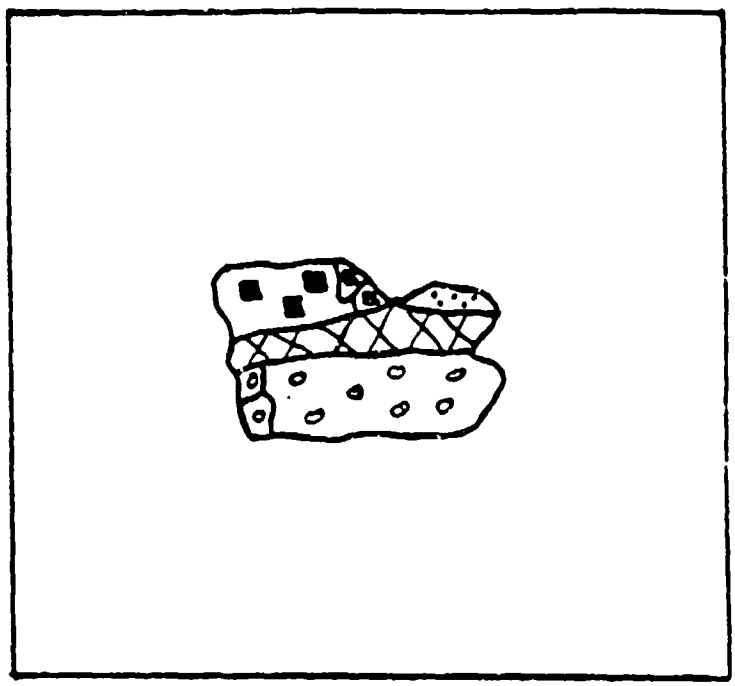


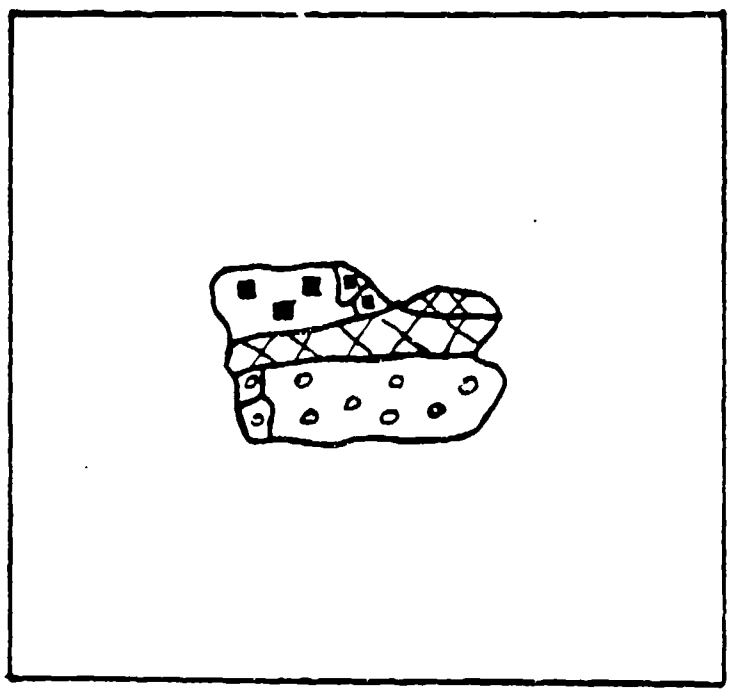
Figure 64. An Example of an Image Understanding Process



(e)

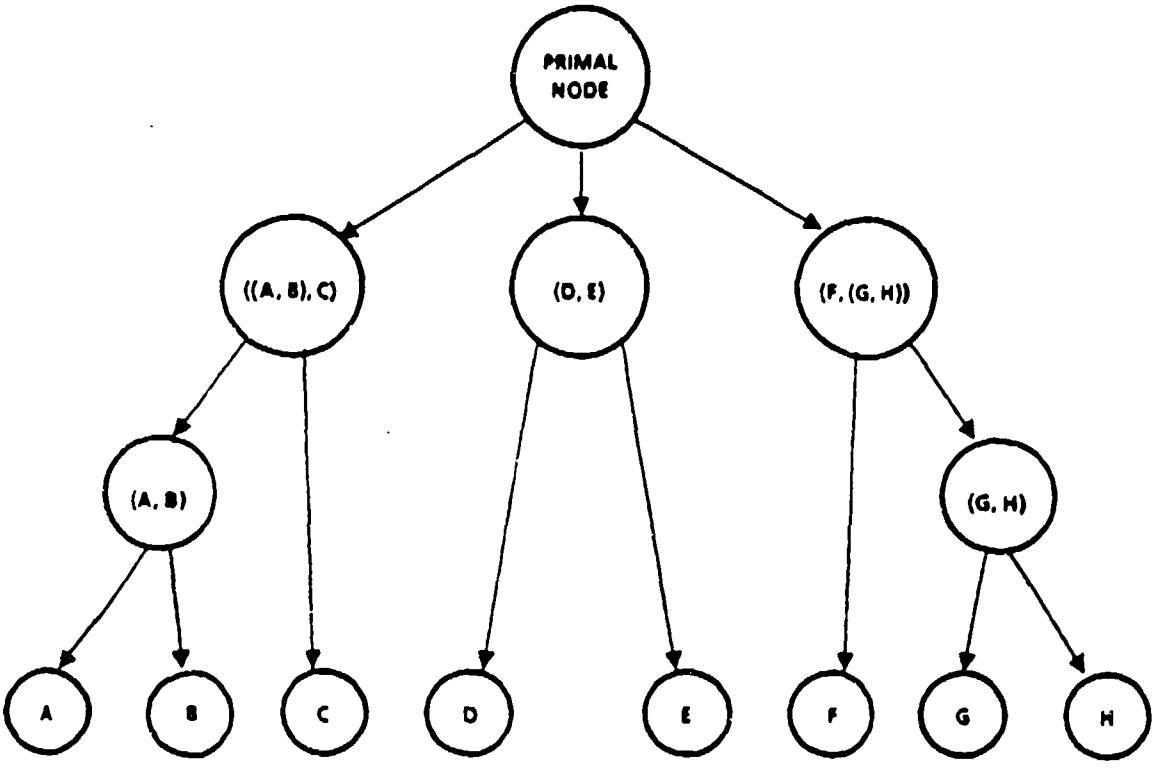


(f)



(g)

Figure 64. An Example of an Image Understanding Process (Continued)



(h)

Figure 64. An Example of an Image Understanding Process (Continued)

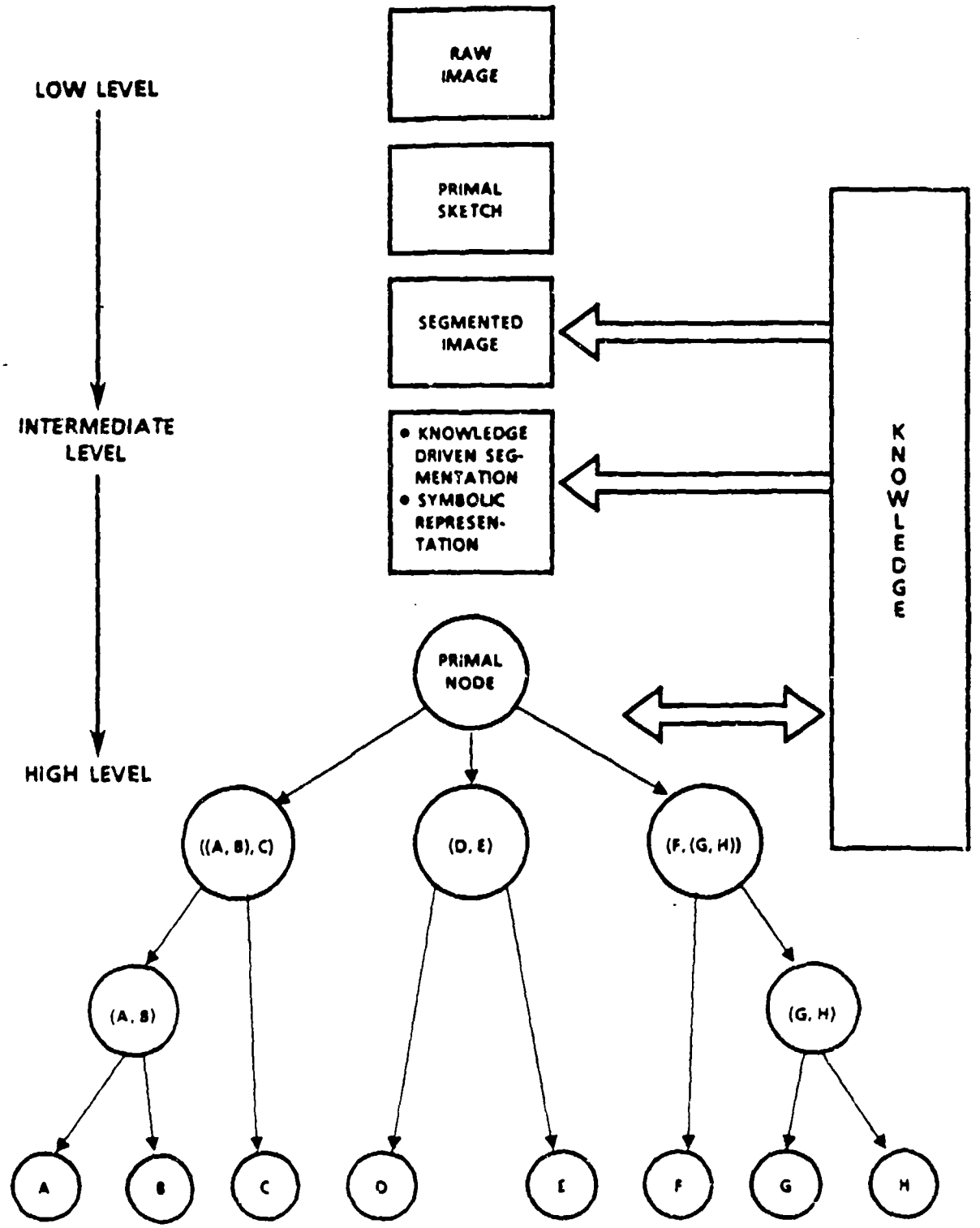


Figure 65. A General Outline for a Scene Analysis Methodology

CHAPTER VII
CONCLUSIONS AND DIRECTIONS FOR FURTHER RESEARCH

This report investigated the various processes used in image understanding by computer as well as the implementation of these processes in a parallel environment. Table 1 provides a general outline of the major topics that have been addressed.

The areas in which image understanding (IU) systems are finding application are growing. For example, expert systems are being designed to aid a trained individual in analyzing different types of imagery from remote sensors, for both commercial and military functions. Another popular example is in the area of robotic devices which are equipped with visual capabilities. Finally, the military is interested in incorporating image understanding methodologies into their imaging systems to create more effective and reliable systems.

Implementation of image understanding algorithms is an important consideration. Many of the algorithms which are well-suited for parallelism have been cited. In general, the algorithms which perform "low-level" operations are those which can be mapped onto a parallel processing architecture. This results from the fact that most of these processes operate on a local neignborhood of pixels. Obviously, many of the promising parallel architectures being designed today, or planned in the future, can have a direct impact in fostering the development of image understanding algorithms for computer vision.

The benefit of using an optical processing environment has also been expounded. Again, many of the applications center around using optics for "low-level" operations. Obviously, the only practical benefit of using optics for image understanding systems is to implement those functions that optical processing performs so well, i.e. correlations and Fourier transform operations. At this time, the general consensus is that an optical correlator is a very powerful processor, and as such, it will be difficult for new, image processing parallel architectures to surpass it. The emphasis, then, should be in using optics where optics has a clear

TABLE 1. IMAGE UNDERSTANDING: PROCESSES AND IMPLEMENTATIONS

	IMAGE UNDERSTANDING PROCESSES		
	LOW	INTERMEDIATE	HIGH

PARALLEL PROCESSING ARCHITECTURE	- IMAGE RESTORATION (SVD)
	- IMAGE ENHANCEMENT (MEDIAN FILTERING)
	- EDGE DETECTION (SOBEL, D2G)
	- EDGE THINNING (NON-MAXIMUM SUPPRESSION)
	- SEGMENTATION
	  - THRESHOLDING
	  - EDGE BASED (RELAXATION)
	  - REGION GROWING
	  - TEXTURE (CO-OCCURRENCE MATRIX)
	  - OPTICAL FLOW (RELAXATION)
		• STATISTICAL PATTERN RECOGNITION
		- CORRELATION
		- FEATURE EXTRACTION (MOMENTS, FOURIER COEFFICIENTS, DISCRIMINANT FUNCTIONS, ETC.)
		- HOUGH TRANSFORM
		• SYNTACTION PATTERN RECOGNITION
		- PARSING
		• 2 1/2 D SKETCH
		- SHAPE FROM SHADING (RELAXATION)
			- GRAPH MATCHING (RELAXATION)
			- HOUGH TRANSFORM

OPTICAL PROCESSING ARCHITECTURE	- IMAGE RESTORATION (SVD)
	- EDGE DETECTION (SOBEL)
	- EDGE THINNING (NON-MAXIMUM SUPPRESSION)
	- SEGMENTATION
	  - THRESHOLDING
	  - EDGE BASED
	  - REGION GROWING
		• STATISTICAL PATTERN RECOGNITION
		- CORRELATION
		- FEATURE EXTRACTION (MOMENTS, DIMENSIONALITY REDUCTION)
		- HOUGH TRANSFORM	

advantage. This can be realized by building hybrid digital/optical systems, where front-end processing is performed by an optical system and intermediate to high level processing is performed by an intelligent back-end digital processing environment. It is important to keep in mind that even though many researchers have analyzed the true effectiveness of optical processing, in terms of accuracy and repeatibility, a proper matching of algorithms to hardware is still needed.

Implementation of the intermediate to high level processes for image understanding in a parallel environment becomes more difficult. The parallel architectures used to implement some of the major algorithms commonly used for intermediate level processing have been described. In general, as the processing gets more symbolic and knowledge oriented and less numeric the further the possibility of going toward a parallel implementation. This is especially true for high level processing.

Most of the research efforts in image understanding has been geared towards three general areas: (1) developing processes which create rich descriptions of a scene for use in higher level processing; (2) extracting intrinsic characteristics from an image; and (3) improving the techniques used in higher level symbolic processing. Higher level processing can be much more efficient if lower level processes produce rich representations of a scene. This has prompted research into developing processes which can extract more information as well as symbolic representations from a scene as early in the processing stages as possible. Many of these requirements imply improving segmentation techniques. Creating rich representations of a scene during early processing is also a result of top-down control, i.e. where low level processes, such as, image enhancement, is monitored by high level requirements.

Extracting intrinsic characteristics from an image, such as depth and orientation, are key processes in image understanding. Some of the approaches taken were discussed in Chapter IV. These characteristics supply valuable information about the visual scene. It also can be used to build three-dimensional geometries of visual surfaces to further aid in scene understanding.

Finally, improvements in higher level processing can be examined in three areas. First, proper characterization of the knowledge base is essential to an image understanding system. An image understanding system has to be designed with an accurate knowledge base. This entails a thorough understanding of the problem domain and the parameters for its solution. Second, where knowledge is incorporated into the image understanding process is a crucial consideration. Generally, inputting knowledge functions in as many places as possible is attractive. However, there is a limit to the overall benefit of such an approach in terms of efficiency, effectiveness, and cost. Third, maintaining a modular system configuration is beneficial in that it allows flexibility in building large and more reliable systems. Modular design is a system concept that can be implemented in the form of rules particular to the high level processing function. Modularity is also attractive in that it promotes creative system design.

A. DIRECTIONS FOR FURTHER RESEARCH

Frustrations with initial attempts at automated image understanding made it clear that vision is so complex a computational task that it is unreasonable, either logically or pragmatically, to conceive it as occuring in a single processing step. As we have shown, visual processing is best thought of as being distributed ove a series of computational stages. This concept was eloquently expressed by Marr.[1] However, one of the major problems in applying this methodology to computer vision has been the lack of a computational approach that unifies all the processing stages. We can refer to this as a unifying computational theory of vision. This has been one of the major reasons why machine vision has been such a difficult problem to solve. Many of the approaches that have been used in the past simply combine a sequence of processing techniques that seemed appropriate for a specific application. However, as the scenes to be analyzed become increasingly complex, this approach quickly breaks down. Researchers have come to the realization that a much higher

level of understanding is needed to construct this computational framework.

In order to develop a complete understanding of the visual processes and representations that should be used in creating a unifying computational theory of vision these elements are needed: (a) a computational theory; (b) the representation and the algorithm; and (c) the mechanism. Marr and his predecessors have already expounded on the mechanistic guidelines. The mechanistic theories are based on biological visual processes which are used to constrain possible theories and guide the development of algorithms. The computational theory is an attempt to create a general theoretical framework which engrosses all levels of processing in order to build rich surface representations for scene understanding. Finally, the representation and the algorithm are the result of transforming a theoretical idea into an engineering concern, i.e., building it.

A processes that attempts to provide a unifying computational theory is relaxation. Relaxation bridges the gap of all processing stages since it can be implemented in all three levels of processing. For example, it has been used for edge detection, edge linking, optical flow, line labeling, shape from shading, semantic net matching, etc. Relaxation also maps nicely into being a local and highly parallel process. This makes it very attractive to parallel implementation, as was discussed. Finally, neuroscientists have conducted numerous experiments which suggest that visual mechanisms also implement local and highly parallel processes. Although relaxation seems very promising it is not the solution in itself. Additional research is needed to discover a computational theory which reconstructs from image primitives a "complete" representation of a scene.

REFERENCES

1. Marr, D., "Early Processing of Visual Information," Phil. Trans. R. Soc., London, B 275, pp. 483-524, 1976.

2. Liu, W., "Two Dimensional Fast Fourier Transform," 1983 IEEE Computer Society Workshop on Computer Architecture for Pattern Analysis and Image Database Management, October 12-14, p. 214-219.

3. Walicki, J. S., Andrews, M., "Parallel Implementation of the LMS Algorithm," Proc. of the 1985 Int. Conf. on Parallel Processing, August 20-23, p. 312-316.

4. Warpenberg, M. R., Siegel, L. J., "SIMD Image Resampling," IEEE Trans. Computers, Vol. C-31, No. 10, October, 1982, pp. 934-942.

5. Athale, R. A., Lee, J. N., "Optical Processing Using Outer-Product Concepts," Proc. IEEE, p. 931-941, July 1984, Vol. 72, No. 7.

6. Abdov, I. E., Pratt, W. K., "Quantitative Design and Evaluation of Enhancement/Thresholding Edge Detectors," Proc. IEEE, p. 753-763, May 1979, Vol. 67, No. 5.

7. Davis, R., Thomas, D., "Systolic Array Chip Matches the Pace of High-Speed Processing," Electronic Design, October 31, 1984, p. 207-218.

8. Cassasent, D., Chen, J., "Nonlinear Local Image Preprocessing Using Coherent Optical Techniques," Applied Optics, Vol. 22, No. 6, p. 808-814, March 15, 1983.

9. Nishihara, H. K., Larson, N. G., "Towards a Real Time Implementation of the Marr and Poggio Stereo Matcher," Image Understanding Workshop, p. 114-120, April 1981.

10. Ibid.

11. See, for example, Reference 8.

12. Ohlander, R., "Analysis of Natural Scenes," Phd. Thesis (Comp. Sci.), Carnegie-Mellon University, Pittsburgh, Pennsylvania, June 1975.

13. Brice, C., Fennema, C., "Scene Analysis Using Regions," Artificial Intelligence, 1, 3, Fall 1970, pp. 205-226.

14. Narayanan, K. A., Peleg, S., Rosenfeld, A., Silberberg, T., "Iterative Image Smoothing and Segmentation by Weighted Pyramid Linking," Univ. of Maryland Computer Science Center, TR-989, December 1980, College Park, Maryland.

15. Politapoulos, A. S., "An Algorithm for the Extraction of Target-Like Objects in Cluttered FLIR Imagery," AESS Newsletter, November, 1980, pp. 23-31.

16. Perkins, W. A., "Area Segmentation of Images Using Edge Points," IEEE Trans. Pattern Analysis Machine Intelligence, 2, January 1980, pp. 8-15.

17. Rosenfeld, A., Hummel, R. A., Zucker, S. W., "Scene Labeling by Relaxation Operations," IEEE Trans. Systems, Man, and Cyber., 6, June 1978, pp. 420-433.

18. Milgram, D., "Region Extraction Using Convergent Evidence," Proc. Image Understanding Workshop, April 1977, pp. 58-64.

19. Narayanan, K. A., Rosenfeld, A., "Image Smoothing by Local Use of Global Information," TR-1006, Computer Vision Lab, Computer Science Center, Univ. of Maryland, College Park, Maryland, February, 1981.

20. Price, K., "Change Detection and Analysis in Multispectral Images," Thesis (Comp. Sci.), Carnegie-Mellon University, Pittsburgh, Pennsylvania, June, 1976.

21. Kelly, M. D., "Visual Identification of People by Computer," AIM-30, Thesis (Comp. Sci.), Stanford University, Stanford, California, July, 1970.

22. Knight, T., MIT AI Lab, private communication.

23. Kushner, T., Wu, A., Rosenfeld, A., "Image Processing on ZMOB," IEEE Trans. Computers, Vol. C-31, No. 10, pp. 943-951, October, 1982.

24. Kuehn, J. T., Siegel, H. J., Grosz, M., "A Distributed Memory Management System for PASM," 1983 IEEE Computer Scoiety Workshop on Computer Architecture for Pattern Analysis and Image Database Management, October 12-14, pp. 101-108.

25. Warde, C., Fisher, A. D., Thackara, J. I., Weiss, A. M., "Image Processing Operations Achievable with the Microchannel Spatial Light Modulator," SPIE, Vol. 252, Smart Sensors II (1980), pp. 25-33.

26. Nevatia, R., Babu, K. R., "Linear Feature Extraction and Description," Computer Graphics and Image Processing, 13, pp. 257-269, 1980.

27. Feldman, J. A., Yakimovsky, Y, "Decision Theory and Artificial Intelligence: I. A Semantics-Based Region Analyzer," Artificial Intelligence, 5, 4, 1974, pp. 349-371.

28. Barrow, H. G, Tenenbaum, J. M, "Experiments in Model-Driven Scene Segmentation," Artificial Intelligence, 8, 3, June 1977, pp. 24-274.

29. Hanson, A. R., Riseman, E. M, "Segmentation of Natural Scenes," CVS, 1978.

30. Horn, B. K. P., Schunck, B. G., "Determining Optical Flow," Artificial Intelligence, 17, pp. 185-23, 1981.

31. Duane, G. S., Venable, S. F., Richter, D. J., Wiederman, A. M., A Production System for Scene Analysis and Semantically Guided Segmentation," Proc. SPIE Conf. on Artificial Intelligence II, Arlington, Virginia, April, 1985.

32. Nazif, A. M., Levine, M. D., "Low Level Image Segmentation: An Expert System," IEEE Trans. Pattern Anal. Mach. Int., Vol. PAMI-6, No. 5, September, 1984.

33. Rice, T. A., Siegel, L. J., "Parallel Algorithms for Computer Vision," 1983 Computer Society Workshop on Computer Architecture for Pattern Analysis and Image Database Management, October 12-14, p. 105-115.

34. Cassasent, D., Fetterly, D., "Recent Advances in Optical Pattern Recognition," SPIE Vol. 456, Optical Computing, 1984, pp. 105-115.

35. Urquahart, R. B., "VLSI Architectures for the Linear Discriminant Function Classifier," 1983 Computer Society Workshop on Computer Architecture for Pattern Analysis and Image Database Management, October 12-14, p. 227-232.

36. Merkle, F., Bille, J., Karfe, T., Dengler, J., Muuss, H., "Hybrid Optical-Digital Image Processing with Optically Generated Component Filters," SPIE, Vol. 236, Optical Systems and Applications, 1980, pp. 102-109.

37. Gorman, F., Clowes, M. B., "Finding Picture Edges Through Collinearity of Feature Points," Proc. Third IJC AI, 1973, pp. 543-555.

38. Ballard, D. H., "Generalizing the Hough Transform to Detect Arbitrary Shapes," Pattern Recognition, Vol. 13, No. 2, pp. 111-122, 1981.

39. Digital/Analog Design, Inc., Technical Report, June, 1985.

40. Hinton, G. E., Sejnowski, T. J., "Optimal Perceptual Inference," Proc. IEEE Computer Vision and Pattern Recognition Conference," Washington, D.C., June 1983, pp. 448-453.

41. Casasent, D., "Symbolic AI Optical Image Pattern Recognition," Technical Proposal, Carnegie-Mellon University, June, 1985.

42. Fu, K. S., Syntactic Pattern Recognition, Prentice Hall, 1982.

43. Chiang, Y. P., Ku, K. S., "A Parallel Earley's Parser and Its Application to Syntactic Image Analysis," 1983 IEEE Computer Society Workshop on Computer Architecture for Pattern Analysis and Image Database Management, October 12-14, pp. 220-226.

44. Horn, B. K. P., Ikeuchi, K., "Numerical Shape From Shading and Occluding Boundaries," Artificial Intelligence, 17, pp. 141-184, 1981.

45. Gibson, J. J., The Perception of the Visual World, Houghton Mifflin, Boston, 1950.

46. Witkin, A. P., "Recovering Surface Shape and Orientation From Texture," Artificial Intelligence, 17, pp. 17-45, 1981.

47. Stevens, K., "Surface Perception from Local Analysis of Texture and Contour," PhD Thesis, Dept. Elec. Eng. and Computer Sci., MIT, Cambridge, Massachusetts, 1979.

48. Faugeras, O., Price, K., "Semantic Description of Aerial Images Using Stochastic Labelling," IEEE Trans. Pattern Analysis Mach. Intell., Vol. PAM-3, No. 6, November, 1981.

49. Hummel, R. A., Zucker, S. W., "On the Foundations of Relaxating Labeling Process," IEEE Trans. Patt. Analy. Mach. Intell., Vol. PAMI-5, No. 3, May 1983, pp. 267-287.

50. Dixit, V., Moldovan, D. I., "Semantic Network Array Processor and Its Applications to Image Understanding," Image Understanding Workshop, October, 1984, pp. 65-71.

ADAPTIVE ARTIFICIAL INTELLIGENCE

AN ORGANIZATIONAL FRAMEWORK FOR COMPARING ADAPTIVE ARTIFICIAL INTELLIGENCE SYSTEMS

Teresa A. Blaxton and Brian G. Kushner

The BDM Corporation
7915 Jones Branch Drive
McLean, Virginia 22102

One of the most interesting topics of investigation in the field of AI is machine learning where systems are being made to automatically "adapt" or learn over time. The following paper presents a common framework for organizing several diverse adaptive AI systems. Nine systems are discussed with regard to: (a) the types of knowledge representations they employ; (b) storage; (c) retrieval mechanisms; (d) conflict resolution principles; and (e) means of adapting knowledge and control structures as a function of changing experience.

There is a new trend evolving in artificial intelligence which directly impacts the work being done on expert systems. More and more, people are becoming unsatisfied with constructing static knowledge bases which are obsolete almost as soon as they are completed and must be updated on a continual basis. The alternative being explored is that of creating systems which adapt with the addition of new knowledge, not only learning new facts but actually changing their own memory organizations and control structures to accommodate the new data more efficiently.[1,2,3] Such systems are potentially more user oriented, more flexible to operate, and are less costly from a software/life-cycle support viewpoint.

Although several researchers have attempted to build adaptive systems (ASs), they have unfortunately done so without the benefit of any guiding theory. Those frameworks that have been offered have not been sufficiently general to encompass the wide variety of systems that have been developed.[4,5,6] Consequently, the literature addresses ASs from multiple perspectives and with inconsistent terminology, making it difficult to compare the accomplishments of these systems with one another. In this article, an organizational structure will be suggested for this body of research, hopefully lending some insights into directions for future development.

This organizational framework has several parts, all focusing on issues relating to the incorporation and utilization of knowledge in ASs. In terms of distinctions between various ASs, the type of knowledge acquired by these systems, as well as how this knowledge is represented, are the aspects most similar to those of traditional expert systems. Other familiar elements of this framework include the retrieval operations used by the AS to access previously stored information, and the mechanisms used to control the processing in the system. But that is where the similarity ends, since both the mechanisms by which incoming knowledge is automatically stored in the system, and the strategies for modifying or adapting the knowledge base and control structure, are by necessity unique features of these adaptive systems. These criteria will be discussed in greater detail in subsequent sections of this paper, where we will in turn (a) establish the need for building ASs; (b) outline a framework of the concerns one might face when trying to build an AS; (c) compare and contrast several already existing ASs; and (d) lay out some guiding principles for building an "ideal" AS.

Problems With Traditional Expert Systems

Many have argued that expert systems are the one area in which the AI enterprise has been truly successful.[8,9] These claims are based upon the proven utility of some computerized expert systems that are used to aid human experts in solving problems within limited application domains.[10,11] Despite these successes, there are still many nontrivial problems associated with building and maintaining expert systems, a few of which will now be enumerated.

To begin, the process of knowledge engineering, whereby the knowledge base of an expert system is acquired, is fraught with difficulties.[12] At worst, the availability of the main expert or experts may be limited or inadequate during the knowledge base construction period. Barring this eventuality, the experts that are available may disagree as to what knowledge to include, leaving the knowledge

engineer at a loss as to how to resolve the dilemma. Perhaps more seriously, the knowledge engineer is faced with the task of obtaining procedural knowledge from the expert who can report only declarative knowledge. That is, the expert can declare that something is true, but cannot accurately describe how to do it.

Aside from problems encountered in the construction of traditional expert systems, a host of difficulties arise once the system is completed. Foremost among these is the challenge of updating the knowledge base to keep it current while maintaining truth values in the system. This becomes particularly acute as the size of the knowledge base increases. The practice of modularizing the knowledge base into separate sections and assigning each to a different knowledge engineer has eased this situation somewhat.[13] However, most would agree that there is still room for improvement, since there are still problems associated with maintaining knowledge consistency and coordination across these modules. Perhaps even a larger concern is that most expert system formulations offer no provision for automatically augmenting the knowledge base and control structure as the need arises. These difficulties, coupled with the often limited domain of applicability of most systems, suggest that investigation of an alternative approach is warranted.

Recent Advances: Adaptive Systems

As was already mentioned, recent work in the area of adaptive systems has led to theoretical advances over traditional expert system formulations.[14] If realized on a large scale, AS would be preferred to expert systems for several reasons. First ASs may be implemented without access to an "expert" per se. That is, the system may start out at a novice knowledge level, and gradually build up to a higher level of expertise through experience and through interaction with some external "knowledge sources." Such a system would be useful in so-called "cutting edge technology" domains where the current knowledge base is small, but expected to grow.

In terms of the user/system interface, ASs may be particularly beneficial in the development of individualized systems where commands accepted by the system are customized for a small set of users. In addition, one might imagine that an AS that has itself progressed from the novice to expert stages would provide more understandable responses to queries made by a novice user than would a traditional expert system.[14]

In a more global sense, ASs are preferable to traditional expert systems for the simple reason that intelligence is dynamic, and in any domain of interest the knowledge and heuristics utilized are bound to change with time. A major amount of effort has been expended in the past to find domains which are suitably narrow and static for building expert systems. In spite of this, the systems that have been developed must still be updated and augmented fairly frequently, and hence require costly, manpower intensive, support tails. One way to avoid this pitfall is to build systems that adapt. Therefore, it is our belief that the performance of the system will be improved to the degree that the system and its knowledge base adapt to the gradual evolution of the domain itself. *

Much of the pioneering work on ASs has been conducted by cognitive psychologists interested in modeling different aspects of human cognition. These researchers have experimented with the application of learning principles to AI systems in domains as diverse as number categorization, puzzle solving, language acquisition, and manipulation of geometric figures. Due to the disparity of these topics, readers of this literature may sometimes find it difficult to apprehend relevant similarities and differences among these systems. The organizational framework presented here is intended to bring a certain order to this chaos by providing points of comparison common to all of these systems, varied though they may be. For purposes of the present paper, the framework will be discussed in regard to only a subset of ASs. ** Before presenting the framework, each of ASs to be discussed will first be briefly described.

Representative Crossection of Adaptive Systems

The first AS to be described is called ACT* and was published by John Anderson.[1] Intended as a comprehensive theory of human cognition, ACT* models such complex activities as rotating mental images, solving geometry proofs, retrieval processes involved in reading, and language acquisition. It is by far the most fully developed system to be described in this article. Since ACT* was designed to model human cognition, it may embody some constraints that are unattractive in the realm of AI applications. Nevertheless it is argued that there is much to learn about building ASs from the study of such a system.

* It has been argued elsewhere that building ASs is not necessarily a worthwhile enterprise in that learning can be a long and iterative process, perhaps requiring more effort than is merited.[33] It is our position, however, that the long term benefits realized from the construction of ASs in dynamic domains will far outweigh any initial startup costs.

** As the reader may notice, the nine systems described in this article constitute only a small number of those programs presented elsewhere as ASs. Some of those other systems have been eliminated from our present discussion because they are better labeled as frameworks for knowledge representation rather than as full blown systems. Still others were omitted because they are not truly adaptive in the sense of having mechanisms responsible for reorganizing memory and control structures. Finally, those deemed too narrow in scope to be of general interest were not included for discussion.[37]

The BACON.5 program written by Langley, Bradshaw, and Simon [15] was intended for a very different purpose. Given numeric data and variable specifications it notices patterns, abstracting out regularities among combinations of variables as "concepts". BACON.5 has discovered the Ideal Gas Law, Ohm's Law, and the law of gravitation, among others. A program somewhat similar to BACON.5 is UNIMEM.[16] It categorizes and makes generalizations from numeric data without using any statistical heuristics such as those employed in the BACON.5 system. For example, given data on areas and populations of states, it categories them into "large" and "small" classes and forms appropriate generalizations about those classes. For instance, it generalizes that small states have small populations without incorrectly inferring that large states necessarily have large populations (e.g., Alaska).

The next three systems are similar to one another in that they all start out with little or no knowledge about a given topic and progress up to an expert level. Kolodner's [17,2,18] Computerized Yale Retrieval and Updating System (CYRUS) learns facts about Cyrus Vance and Edmund Muskie's terms as Secretary of State. The Integrated Partial Parser, IPP, learns about international terrorism from newspaper articles. Finally, a follow-up to IPP called RESEARCHER [19] learns and answers questions about patent abstracts.

For purposes of the present paper, the most notable aspects of all of these systems are that they have mechanisms for organizing and updating their knowledge bases such that interesting generalizations and discriminations result. For example, after learning about a number of instances in which kidnap victims in Italy happened to be businessman, IPP is able to generalize that a new (unidentified) Italian who is kidnapped is likely to be a businessman. IPP does not make the error of the converse, however, which would be to infer that a given kidnapping took place in Italy simply because the victim is a businessman.

An example of an AS in a very different domain is the Blocks World program, first introduced by Winograd [20] and then updated by Winston.[21] This system learns about various possible configurations of a set of geometric figures. For example, Winograd's [20] system can remember sequences of moves for any given object in the blocks world and infer procedures necessary for those sequences to have been performed, even though that information was not explicitly stored.

The last two ASs learn rules about how to solve some problem. The highly acclaimed Meta-DENDRAL [22] is a program which sits over the expert system DENDRAL and learns cleavage rules used by mass spectrometers. The second version of the Strategic Acquisition Governed by Experimentation system, SAGE.2 [3] learns rules for solving the Tower of Hanoi puzzle task. Starting out with a small number of heuristics this program quickly acquires the knowledge necessary to solve the puzzle in the minimum number of moves.

Having read these descriptions the reader should now have an appreciation for how diverse these ASs really are. Nevertheless, it is argued that an organizational framework may be used to view these systems which will allow their simultaneous comparison on a number of dimensions. That framework will now be presented.

Framework for Comparing Adaptive Systems

The ASs [redacted] just described may be compared with regard to the following features: (a) the types of representation schemes employed; (b) the mechanisms by which incoming knowledge is stored in the system; (c) retrieval operations used to access previously stored information; (d) mechanisms used to control information processing in the system; and (e) strategies for adapting the knowledge base and control structure to accommodate the changing environment. In this section, our set of ASs will be discussed with regard to each of these features. Following this, an evaluation of the different designs employed in building these ASs will be presented.

A. Type of Knowledge Representations Used

1. Production Rules

The first type of representation listed in Table 1 is the production rule. Production rules are if-then or condition-action pairs which invoke a particular action to be carried out when the contents of working memory match a specified condition. By this description it is clear that production rules embody procedural knowledge, but they are closely tied to declarative knowledge as well. Consider the following example of a production rule in the BACON system: [23]

> If you see a number of descriptions at Level L in which the dependent variable (D) has the same number value (V),
>
> Then create a new description at level L+1 in which the value of D is also V and which has all conditions common to the observed descriptions.

This rule results in the creation of another production, which will, in some sense, represent declarative knowledge within its own conditions. When these new conditions are matched by the contents of working memory, the new rule will be eligible for implementation. Production rules are used in ACT*, BACON.5, Meta-DENDRAL, and SAGE.2.

2. Memory Organization Packets (MOPs)

Following in a tradition somewhat similar to that of Minsky's frames, [24] Schank and Abelson's scripts, [25] and Schank's dynamic memory, [26] UNIMEM, CYRUS, IPP, and RESEARCHER all use memory organization packets, or MOPs. MOPs are a type of semantic network within which knowledge is

arranged hierarchically by topic.[27] Information that constitutes a specific exception to the theme of the MOP is given a separate index, or label, which may be used to locate it more quickly during the retrieval process. When two or more elements are organized under one index, a new sub-MOP is formed, thus preserving the modularity of the memory network.

3. Propositions

Propositions have become a fairly common means of abstract knowledge representation in AI systems. They preserve semantic relationships between arguments or objects. For instance, the proposition "(kiss Sue Bill)" preserves the relation "kiss" between the arguments "Sue" and "Bill". Notice that the proposition's structure is independent of information order.[1] That is, "(kiss Sue Bill)" does not encode the difference between "Sue kissed Bill" and "Bill was kissed by Sue". Only the semantic relation is represented. Propositional representations are used both in ACT* and the Blocks World system.

4. Temporal Strings

Temporal strings are a type of representation used in the ACT* system to maintain order information. Knowledge about order is important to many tasks, one of which is the analysis of linguistic information. Temporal strings provide a more efficient means of storing order than do propositions, but cannot be used in the place of propositions since they do not incorporate information about meaning as effectively.

5. Spatial Images

This final type of representation preserves the configuration of elements in a spatial array. This construct is used in the ACT* system to model pattern recognition behavior in humans. The important point to note here about spatial images is that they preserve only information about the relative physical positions of objects in an array, and not necessarily information about what these objects actually look like.

B. Information Storage Strategies

Having established the tools for representing knowledge in our set of ASs, it is now of interest to explore the strategies used to store new information in these systems.

As shown in Table 2, one strategy for adding new information to the system is to learn every new stimulus as it is presented. This simple approach is used in the UNIMEM, CYRUS, Blocks World, and Meta-DENDRAL systems. Another alternative is to be more discriminating and store new information in the network only after it has been shown to have some importance, for instance after it has occurred several times. The ACT*, BACON.5, IPP, RESEARCHER, AND SAGE.2 systems all employ this type of approach.

Once the decision has been made to add new information to the system, the question arises as to how it will be stored in relation to already existing memory elements. Of course the new data could simply be added randomly. However, a more strategic approach is to store related items "near" one another in the network. The ACT*, UNIMEM, CYRUS, IPP, and RESEARCHER systems all employ this design, linking together items which are semantically or contextually related.

This method of storage serves two useful purposes. First, provided one uses the right type of knowledge representation, it will not be necessary to explicitly store all semantic features associated with every new element since some of those may already be represented in nearby (subsuming) structures. Second, retrieval of information is facilitated when memory is organized thematically. that is, one need only get to the right area in memory and look there rather than exhaustively searching the entire network.

Another useful tactic, implemented in UNIMEM, CYRUS, IPP and RESEARCHER, is to mark new elements with special indices which denote the way in which their themes differ from those of their "parent" nodes in the network. This approach has the same advantages as storing related items near one another in memory. It is particularly useful in retrieval as will be illustrated in the next section.

C. Retrieval Mechanisms

The three types of retrieval mechanisms employed by our set of ASs are listed in Table 3. The first of these, pattern matching, is a method whereby an item is retrieved from memory if the physical features of its representation match those of some retrieval cue. Pattern matching is commonly used in production systems where rules are selected for use based on the match between their conditions and the elements in working memory. The implementation of this mechanism usually involves exhaustive memory search. The ACT*, BACON.5, Blocks World, Meta-DENDRAL, and SAGE.2 systems all employ pattern matching.

A very different approach to retrieval is to search the network by traversing indices associated with individual categorical memory structures. Recall that the MOPs used to represent knowledge in the UNIMEM, CYRUS, IPP, and RESEARCHER systems, [redacted] There are indices which denotes the thematic content of an MOP and other indices which tag elements involving exceptions to these themes. Retrieval begins with an index which is compared to other indices in the network. When a match is found, the memory elements subsumed underneath that index are searched. The result is that the scope of the retrieval process is limited to only those areas of memory which are the most relevant, a more efficient strategy than exhaustive search.

Another way of facilitating economic search is through spreading activation, as implemented in ACT*. At any given time, the memory elements that match the contents of working memory are temporarily activated, and this activation spreads to "nearby" connected elements in the network. Since the storage strategy in ACT* is to form connections among related items as they are learned, the nearby items that get activated will also be related to the contents of working memory. That is, all activated elements will be potentially relevant to the current context. Search is quite efficient since the retrieval process is directed only to the areas of the network that are activated, rather than to the network as a whole.

D. Principles of Control and Conflict Resolution

As might be expected, the search mechanisms just described often result in the retrieval of more than one acceptable memory element. The problem then arises as to how to choose among the potential candidates, selecting the most appropriate one for instantiation. A set of principles used to resolve these conflicts is presented in Table 4.

The most obvious means of deciding among potential elements is to choose the one which most closely matches the retrieval cue (or contents of working memory). this strategy is adopted in all of the ASs we are considering. In practice, however, one might find that as systems get more and more complex, this simple tactic will not always work well. That is, there may be several elements which match to the same degree, in which case further means for choosing among elements must be provided.

The ACT* and SAGE.2 systems employ several strategies for resolving these conflicts which rely on analysis of features of the operators themselves. For instance, each operator in these ASs has some strength associated with it which is either increased or decreased after each instantiation, depending on the outcome. Competing operators are then selected depending upon their relative strengths or histories of being useful. The field of potential candidates can be narrowed further using principle called data refractoriness , .eby no one operator can serve in two patterns simultaneously. Finally, the ACT* program relies on a principle of specificity whereby the most specific of two operators which match equally well will be chosen. For instance, if the pattern "barn" were being matched and the two elements "ba" and "bar" were competing, "bar" would be chosen since it is the most specific.

In addition to relying on traits of the individual operators to resolve conflicts, system behavior may be controlled in both ACT* and SAGE.2 by contextual constraints. Context is used to govern choice of operators in SAGE.2 through the "recency of use" rule. That is, the most recently used operator is chosen over others on the assumption that it must be the most relevant to the problem at hand. This choice will have nothing to do directly with the strength of the operator or whether the pattern is already represented elsewhere.

Perhaps a more interesting mechanism is used in the ACT* program. Problem solving in ACT* is goal-driven. That is, a large task having one ultimate goal can be broken down into several subtasks, each having a goal of its own. The current goal of interest is represented in working memory and, as such, disallows the instantiation of any operators not directly related to its completion. This mechanism greatly reduces the field of potential candidates from the start, thus eliminating many conflicts that might otherwise arise.

E. Mechanisms for Adaptation

Thus far the types of mechanisms described are not in any way peculiar to systems under consideration here. For instance, any traditional production system has to have some means of retrieving information from memory (usually pattern matching), and some way to resolve conflicts among competing memory elements (usually degree of match and specificity). The trouble with these typical expert systems, however, is that when they "learn" they do so simply by adding new facts or rules. These additions are, for the most part, made without regard to the integration and reorganization of this new information with existing knowledge.[28]. What sets adaptive systems apart from traditional expert systems is their ability to modify their own knowledge an control structures with experience in some meaningful manner. The ways in which adaptation occurs are listed in Table 5.

The most common way of incorporating experience into the system is to strengthen operators that have either been presented a number of times or have somehow proven useful in the past. In fact, every AS in our set of nine except Blocks World employs this method. However, the converse of this strategy, which is to weaken an operator whose instantiation has produced undesirable results, is not as popular. Only the ACT* and SAGE.2 systems employ this tactic. A method more drastic than weakening the strength of an operator is to remove it from the network altogether when if ceases to be appropriate for current applications. This is sometimes done with production rules in ACT*, but only very conservatively.

In addition to changing the relative strengths and weaknesses of operators in the system, it is possible to actually create new ones using knowledge embedded in previously existing representations. In particular, generalization involves the formation of new elements which embody common features of several representations already in the network. The new operator applies in more cases since it is less specific than its predecessors. This mechanism is implemented in every AS except Meta-DENDRAL and SAGE.2.

In contrast to generalization, new representations may be created through the process of discrimination. Discrimination occurs when extra delimiting features are added to an operator which restrict the contexts in which it can apply. In other words, discrimination creates constructs that are special cases of already existing ones. This is the type of process that occurs in CYRUS, for instance, when an index is added beneath the top level of an MOP to signify that an element contains information in exception to the theme of the parent MOP. Although potentially quite powerful, this strategy is employed only in the ACT* and CYRUS systems.

The discussion up to this point has focussed on the comparison of our nine ASs with regard to the types of knowledge they acquire, along with the types of representations, storage mechanisms, retrieval strategies, control principles, and means of adaptation used by each. Now that the ASs have been couched in terms of this organizational framework, it is of interest to determine whether any guiding principles may be used in conjunction with this framework to help researchers both (a) build better ASs and (b) evaluate our nine ASs in relation to one another.

Building the "Ideal" Adaptive System

The remainder of this paper will outline some general guidelines for building what we call the "ideal" AS. These guidelines will be cast in terms of the organizational framework already presented. It is hoped that these principles will not only aid in the design of future ASs, but might serve as a set of metrics by which to evaluate already existing systems.

First it is our feeling that there is no way to select the "best" representation a priori. In fact some might argue that, as far as the outward behavior of the system is concerned, any deficiencies associated with the choice of representation structure may be compensated for in terms of the procedures implemented to operate on those structures.[29] We will concede that some representations may lend themselves to certain types of problems more easily than others, although this will be a matter of convenience.

Having dispensed with the question of representation, we will now proceed with recommendations concerning the other issues of storage, retrieval, control, and adaptation. After reading through the earlier comparisons of the ASs on each of the dimensions just listed, the reader may feel that a "more is better" rule is applicable. That is, it might appear that variety is the key, with the better systems being those which incorporate several different types of capabilities in each of these areas. Although this is true to a certain extent, it is not necessarily the best prescription for success.

Rather than a "more is better" methodology, we advocate that researchers adopt an approach whereby a combination of both top-down and bottom-up strategies are implemented in the system design. Bottom-up strategies are those for which properties of individual operators are important, whereas top-down strategies involve overall system behavior. Perhaps this concept is best illustrated with an example.

In terms of storage mechanisms, a top-down strategy is to store related items near one another in the network. This type of strategy impacts directly on the global organization and performance of the system. Similarly, the approach of storing items in a hierarchial organization such that concepts at a given level subsume those below and are subsumed by those above will later affect global retrieval operations.

In contrast, a bottom-up storage strategy is to learn every new stimulus as it is presented, regardless of its semantic content--that is, regardless of its eventual position in the overall memory network. This strategy has not been employed as much as it could have been, probably due to the semantic primacy bias in the human memory literature.[30] The traditional wisdom has been that people primarily remember semantic content in lieu of lower level information involving physical features of stimuli. Recent experiments on human memory have shown, however, that we do indeed remember this low level information, often better than the semantic content.[31] To the degree that the human is accepted as a useful model for designing intelligent systems, the importance of bottom-up storage strategies should not be ignored when building ASs.

Assessing the relative merit of combinations of storage mechanisms of the ASs from Table 3, it may be seen that only the UNIMEM and CYRUS systems employ both top-down and bottom-up strategies. Specifically, both are designed to learn every new stimulus as it is presented and to store related items near one another in memory. In addition, both of these systems use the MOP representation from which a hierarchical organization is created.

Applying the top-down/bottom-up distinction to retrieval mechanisms, it may be argued that memory search aided by either index traversal or spreading activation is top down since these mechanisms are implemented with regard to the memory network as a whole. On the other hand, pattern matching is a bottom-up retrieval tactic since it depends only on characteristics of individual operators. Again, research on human memory has shown that both bottom-up and top-down processes play important roles in the determination of retrieval performance [32] lending credence to the assertion that this combination might be useful in the design of ASs. In Table 4 we see that the only system employing both top-down and bottom-up retrieval strategies is ACT*. Elements may be retrieved from memory in ACT* using either spreading activation or pattern matching.

As with storage and retrieval, there are principles of conflict resolution which embody both top-down and bottom-up features. For example, a top-down strategy for delimiting the field of competing operators is to impose a goal hierarchy on the system. The choice of priorities rather than the traits of any individual operators. A similar argument may be made for the strategy of choosing the most recently used operators over others that match as closely.

All other conflict resolution principles discussed earlier involve bottom-up analysis. These include degree of match, specificity, data refractoriness, and operator strength. Notice that all of these require that features of individual operators dictate which will be selected for implementation as opposed to overall contextual constraints.

Looking to Table 5 again, it may be seen that the only two systems which employ top-down and bottom-up strategies for controlling choice of operators are ACT* and SAGE.2. Both use several bottom-up strategies. Of greater interest is that the top-down approach to control in ACT* is implemented in a goal hierarchy, whereas a recency criterion serves as a context mechanism in SAGE.2.

The final dimension to be considered from the organizational framework is the manner in which systems are allowed to adapt. Of the mechanisms presented earlier, adaptation through generalization and discrimination are top-down in nature because they occur only when similarities or differences among several structures are noticed simultaneously in one context. On the other hand, the strengthening, weakening, and unlearning of individual elements all affect only one structure at a time. Since they do not directly impact on the status of other contextually-related elements, these strategies are classified as being bottom-up. All of the ASs invoke both top-down and bottom-up adaptation strategies except Meta-DENDRAL and SAGE.2 which employ only bottom-up procedures.

Conclusions

The above discussion focused on developing an organizational framework for comparing adaptive systems. These systems are of interest relative to traditional expert systems in that they (a) do not have to be updated by hand as knowledge in the domain of interest changes; (b) can start out at the novice level without explicit access to a human expert; (c) are useful in "cutting edge" domains in which few, if any, experts exist; and (d) might provide better interfaces for novice users since their own memory structures have evolved from the novice level. Nine systems were discussed with regard to an organizational framework for comparing ASs. This framework was used to compare the systems on such dimensions as the type of knowledge acquired by each and representations used; storage and retrieval strategies, control mechanisms, and methods used for adaptation. [redacted] Based on our experience with the latest generation of expert systems, we have recommended that ASs be designed using a combination of top-down and bottom-up mechanisms in each of these areas.

The development of this framework for ASs systems may also have several additional benefits. As the most near term example, this framework can aid in our understanding of AI system performance. This may allow researchers to address some of the perennial AI questions, such as how one measures the robustness of a given AI system, what are valid benchmarks for AI systems, and how an AI system can be designed for ease of testing. On this latter point, research in AS methodologies could expedite the rapid prototyping of AI systems, in a manner analogous to the techniques used in the electronics industry. Finally, continued development of ASs, particularly through rapid prototyping, can lead to a faster incorporation of AI systems into the marketplace.

Acknowledgments

The authors would like to thank Drs. L. H. Reeker, R. A. Geesey, C. B. Friedlander and C. T. Butler for their timely insights and constructive criticism throughout the development of this effort. This work was supported in part by DARPA under contract number MDA903-85-C-0056 and by DARPA/AFOSR under contract number F49620-86-C-0036.

REFERENCES

(1) Anderson, J. R. (1983) The architecture of cognition, Harvard University Press: London.

(2) Kolodner, J. A. (1983a) Maintaining organization in a dynamic long term memory. Cognitive Science, 7, 243-280.

(3) Langley, P. (1985) Learning to search: From weak methods to domain-specific heuristics. Cognitive Science, 9, 217-260.

(4) Bundy, A., Silver, B., & Plummer, D. (1985) An analytical comparison of some rule learning programs. Artificial Intelligence, 27, 137-181.

(5) Carbonell, J. G. (1983) Learning by analogy: Formulating and generalizing plans from past experience. In Carbonell, J. G., Michalski, R. S., & Mitchell, T. M. (Eds.) Machine learning: An artificial intelligence approach. Tioga Publishing Co., Palo Alto, CA.

(6) Michalski, R. S. (1983) A theory and methodology of inductive learning. In Carbonell, J. G., Michalski, R. S., & Mitchell, T. M. (Eds.), Machine learning: An artificial intelligence approach. Tioga Publishing Co., Palo Alto, CA.

(7) Rychener, M. D. (1983) The instructible production system: A retrospective analysis. In Carbonell, J. G., Michalski, R. S., & Mitchell, T. M. (Eds.), Machine learning: An artificial intelligence approach. Tioga Publishing Co., Palo Alto, CA.

(8) Bobrow, D. G., & Hayes, P. J. (1985) Artificial intelligence--Where are we? Artificial Intelligence, 25, 375-415.

(9) Hayes-Roth, F. (1985) Engineering systems of knowledge: The great adventure ahead. In (K. N. Karna, Ed.) proceedings of the Expert Systems in Government Symposium, pp. 678-692.

(10) Pople, H. (1982) Heuristic methods for imposing structure on ill-structured problems: The structure of medical diagnostics. In P. Szolovits (Ed.) Artificial intelligence in medicine. Westview Press Co.

(11) Shortliffe, E. (1976) Computer-based medical consultations: MYCIN. American Elsevier, New York.

(12) Hayes-Roth, F. (1984) The knowledge-based expert system: A tutorial. IEEE, September, pp. 11-28.

(13) Frosher, J. N., & Jacob, R. J. K. (1985) Designing expert systems for ease of change. In Proceedings of the IEEE Computer Society Expert Systems in Government Symposium, K. N. Karna (Ed.), Computer Society Press.

(14) Riesbeck, C. K. (1984) Knowledge reorganization and reasoning style. In M. J. Coombs (Ed.) Developments in expert systems, Academic Press: London.

(15) Langley, P., Bradshaw, G. L., & Simon, H. A. (1981) BACON.5: The discovery of conservation laws. In proceedings of the 7th International Joint Conference on Artificial Intelligence, 121-126.

(16) Lebowitz, M. (1985) Categorizing numeric information for generalization, Cognitive Science, 9, 285-308.

(17) Kolodner, J. A. (1983a) Maintaining organization in a dynamic long term memory. Cognitive Science, 7, 243-280.

(18) Kolodner, J. A. (1983c) Retrieval and organizational strategies in conceptual memory: A computer model. Lawrence Erlbaum Associates, Hillsadale, N.J.

(19) Lebowitz, M. (1986) An experiment in intelligent information systems: RESEARCHER. Unpublished manuscript.

(20) Winograd, T. (1972) Understanding natural language. New York: Academic Press.

(21) Winston, P. (1975) Learning structural descriptions from examples. In P. H. Winston (Ed.), The psychology of computer vision, New York: McGraw-Hill.

(22) Buchanan, B. G., Smith, D. H., White, W. C., Gritter, R. J., Feigenbaum, E. A., Lederberg, J., & Djerassi, C. (1976) Applications of artificial intelligence for chemical inference XXII: Automatic rule formation in mass spectormetry by means of the META-DENDRAL program. Journal of the American Chemical Society, 98, 6168-6178.

(23) Langley, P., Bradshaw, G. L., & Simon, H. A. (1983) Rediscovering chemistry with the BACON system. In Carbonell, J. G., Michalski, R. S., & Mitchell, T. M. (Eds.), Machine learning: An artificial intelligence approach. Tioga Publishing Co., Palo Alto, CA.

(24) Minsky, M. (1975) A framework for representing knowledge. In P. Winston (Ed.) The psychology of computer vision. McGraw-Hill.

(25) Schank, R. C., & Abelson, R. (1977) Scripts, plans, goals, and understanding. Hillsdale, NJ: Erlbaum.

(26) Schank, R. C. (1982) Dynamic memory: A theory of reminding and learning in computers and people. Cambridge University Press: London.

(27) Chandrasekeran, B., & Mittal, S. (1984 Deep versus compiled knowledge approaches to diagnostics problem solving. In M. J. Coombs (Ed.) Developments in expert systems, Academic Press: London.

(28) Kolodner, J. A. (1984) Towards and understanding of the role of experience in the evolution from novice to expert. In M. J. Coombs (Ed.) Developments in expert systems, Academic Press: London.

(29) Anderson, J. R. (1978) Arguments concerning representations for mental imagery. Psychological Review, 85, 249-277.

(30) Sachs, J. S. (1967) Recognition memory for syntactic and semantic aspects of connected discourse. Perception and Psychophysics, 2, 437-442.

(31) Kolers, P. A. (1976) Reading a year later. Journal of Experimental Psychology: Human Learning and Memory, 2, 554-565.

(32) Roediger, H. L., & Blaxton, T. A. (in press) Retrieval modes produce dissociations in memory for surface information. In D. S. Gorfein and R. R. Hoggman (Eds.) Memory and cognitive processes: The Ebbinghause Centennial Conference. Hillsadale, N.J.

(33) Buchanan, B. G., Barstow, D., Bechtal, R., Bennett, J., Clancey, W., Kulikowski, C., Mitchell, T., & Waterman, D. A. (1983) Constructing an expert system. In F. Hayes-Roth, D. A. Waterman, & D. B. Lenat (Eds.) Building expert systems. London: Addison-Wesley.

(34) Brachman, R. J., & Schmolze, J. G. (1985) An overview of KL-ONE knowledge representation system. Cognitive Science, 9, 171-216.

(35) Feigenbaum, E. A., & Simon, H. A. (1984) EPAM-like models for recognition and learning. Cogntive Sciences, 8, 305-336.

(36) Lenat, D. B. (1983) The role of heuristics in learning by discovery: Three case studies. In Carbonell, J. G., Michalski, R. S., & Mitchell, T. M. (Eds.), Machine learning: An artificial intelligence approach. Tioga Publishing Co., Palo Alto, CA.

(37) Quinlan, J. R. (1983) Learning efficient classification procedures and their application to chess and games. In Carbonell, J. G., Michalski, R. S., & Mitchell, T. M. (Eds.), Machine learning: An artificial intelligence approach. Tioga Publishing Co., Palo Alto, CA.

Table 2

STRATEGIES FOR STORING INFORMATION

	Learn every new stimulus	Learn only "important" stimuli	Store related items near one another	Index for ease of search
ACT*		I	I	
BACON.3		I		
UNIMEM	I		I	I
CYRUS	I		I	I
IPP		I	I	I
RESEARCHER		I	I	I
BLOCKS WORLD	I			
META-DENDRAL	I			
SAGE.2		I		

Table 3

MECHANISMS USED FOR INFORMATION RETRIEVAL

	Pattern Matching	Index Traversal	Spreading Activation
ACT*	I		I
BACON.3	I		
UNIMEM		I	
CYRUS		I	
IPP		I	
RESEARCHER		I	
BLOCKS WORLD	I		
META-DENDRAL	I		
SAGE.2	I		

TABLE 1

CONSTRUCTS USED FOR KNOWLEDGE REPRESENTATION

	Production rules	AVPs	Propositions	Temporal strings	Spatial images
ACT*	I		I	I	I
BACON.3	I				
UNIMEM		I			
CYRUS		I			
IPP		I			
RESEARCHER		I			
BLOCKS WORLD			I		
META-DENDRAL	I				
SAGE.2	I				

Table 4

PRINCIPLES OF CONTROL

	Degree of Match	Operator Strength	Data Refractoriness	Specificity	Recency of use	Goal Dominance
ACT*	I	I	I	I		I
BACON.3	I					
UNIMEM	I					
CYRUS	I					
IPP	I					
RESEARCHER	I					
BLOCKS WORLD	I					
META-DENDRAL	I					
SAGE.2	I	I	I		I	

Table 5

MECHANISMS FOR ADAPTATION

	Strengthening	Weakening	Unlearning	Generalization	Discrimination
ACT*	X	X	X	X	X
BACON.5	X			X	
UNIMEM	X			X	
CYRUS	X			X	X
IPP	X			X	
RESEARCHER	X			X	
BLOCKS WORLD				X	
META-DENDRAL	X				
SAGE.2	X	X			

ATTENTIVE ASSOCIATE ARCHITECTURE

THE BDM CORPORATION

ATTENTIVE ASSOCIATIVE ARCHITECTURES AND THEIR IMPLICATIONS TO OPTICAL COMPUTING

R. A. Athale, C. B. Friedlander, and B. G. Kushner

The BDM Corporation

7915 Jones Branch Drive

McLean, VA 22102

Abstract

The concept of Attentive Associative Processing trades the spatially delocalized and self-organizing features of traditional Associative Architectures for improved hardware efficiency and increased speed of adaption. Optical computing may prove to be ideally suited to implement these architectures.

Introduction

Optical computing systems, which hold the promise of massive parallelism in computation and interconnects, have attracted the attention of researchers for the last 30 years. The earliest systems proposed and demonstrated used the ability of a simple and passive (non-power consuming) component like a convex lens to perform a complex 2-D Fourier transform extremely rapidly. This early work in optical computing/processing was directed toward processing of synthetic aperature radar data, matched filter detection of images, image restoration, and radar and sonar signal processing. As this work has matured and has transitioned to advanced development and deployment, the optical computing research community has turned its attention to tackling a wider variety of problems. Current research is evolving along two distinct directions: (1) Toward optical logic devices capable of extremely high speed and/or highly parallel operation, (2) Towards analog systems that use the massive parallelism in arithmetic operations and interconnects for novel architectural configurations. In the last two years, the optical computing community has been strongly motivated to study the work in human cognition, and apply the architectural concepts developed in modeling neural networks. This new direction has been driven by the conviction that the human brain derives its astounding power for cognition and perception from a massively parallel and densely interconnected network of relatively simple and slow components - i.e. neurons, and that these are exactly the characteristic strengths of optical systems as conceived in the second direction mentioned above. The discipline of neural network modeling is in its infancy relative to a more developed field like signal processing and ditigal computing, and there is a notable lack of universally accepted principles to guide the architectural development. Nonetheless, in the past 40 years a significant amount of interdisciplinary research has been carried out in this field and has generated valuable insights into the operation of a human brain. Most notably, the work of Grossberg[1], Kohonen[2], and Hopfield [3] has influenced the work in optical computing initiated by Psaltis and Farhat[4] and Fisher, Giles, and Lee[5]. The growth of research in the field of optical associative processing can be judged by the large number of presentations at the Associative Memories and Optics Symposium at the 1985 Annual Meeting of the Optical Society of America as well as at the Los Angeles Symposium of SPIE.

In this paper, we will describe that portion of our work in the area of optical associative processing that is a departure from the work referred to earlier. In the earlier works, the associative storage was based mainly on the correlation matrix of the data vectors. We propose an associative memory model that readily allows a change in the strengths of stored states, corresponding to a shift in attention[6]. In this paper we discuss the implications of attentive associative architectures to optical computing. We describe the mathematical formulation and optical implementation of attentive associative memory and its relation to the conventional associative memory, and we discuss how these concepts can be transferred to different application areas of optical computing, e.g. relational data base architectures and expert systems.

Attentive associative memory

The simplest model of an associative memory designed to store N-dimensional column vectors, is obtained in two steps: The first step is the recording of the set of P input

vectors in an N x N memory matrix via the outer product operation between the input vectors, and the second step is retrieving the data vector from an incomplete and/or noisy version of the vector itself via a vector-matrix multiplication. These two steps are described mathematically in the following equations [2]:

$$M = \sum_i \underline{v}^{(i)} \underline{v}^{(i)T} \qquad \text{RECORDING}$$

$$\underline{v} = M \underline{v}' \qquad \text{RETRIEVAL} \quad [i]$$

$$v_j = \sum_k M_{jk} v'_k$$

where $\underline{v}$ is an N-dimensional column vector, M is an N x N matrix, $\underline{v}'$ is the imperfect recall vector, and $\underline{v}^{(i)}$ is one of the stored vectors. The last part of eq.[1] can be rewritten by a substitution for values of M_{jk} derived from the first part of eq.[1]:

$$v_j = \sum_k \left(\sum_i v_j^{(i)} v_k^{(i)} \right) v'_k \qquad [2]$$

The attentive associative memory formulation can be obtained by changing the order of summation eq.[2] and inserting a nonuniformly nonlinear operation after the first summation. The resultant equation is given below:

$$v_j = \sum v_j^{(i)} F_1(i) \left(\sum_k v_k^{(i)} v'_k \right) \qquad [3]$$

This equation states that the imperfect input vector $\underline{v}'$ is first compared to all the stored vectors in parallel via an inner product, the resultant scalar is transformed using a channel-dependant nonlinearity, F (i), and then used as a coefficient in a linear superposition of the corresponding stored vectors. The output is an estimate of the stored vector that is closest to the input vector, $\underline{v}'$. The nonlinear operation, F (i), allows one to suppress spurious correlations and to emphasize the similarity of the input with a selected vector (i.e. focussing attention on that particular vector).

This basic model of associative memory can be modified in numerous ways, some of them discussed in Ref. 2. In Ref. 3, Hopfield suggests an iterative procedure where the estimate of the retrieved vector ($\underline{v}$ in eq. [1]) is used to calculate an improved estimate. In that work, the data vectors were chosen to be binary, and this knowledge was used in hardclipping the retrieved vector before feeding it back to the system. Ref. 4 discusses the optical implementation of the Hopfield model via an optical vector-matrix multiplier with feedback and a threshold nonlinearity in the feedback loop. The same procedure can be applied to the attentive associative memory model described in eq. [3] to improve the quality of retrieval.

In Ref. 4 an extension of the Hopfield Model to storage of images was proposed. Since images are 2-D matrices, their outer products result in a 4-D tensor (corresponding to the recording step in eq. [1]). To facilitate the realization of this tensor, Ref. 4 proposed an optical system equivalent to eq. [3] in that it also performs the inner product between the images before forming the linear superposition of the stored images. The system, however, did not involve the nonlinear step contained in eq. [3]. A recent paper by Soffer et al discusses a holographic implementation of the associative memory for storing images, in which the correlation between the input and the stored images was subject to a nonlinear operation[7]. That system did not contain provision for a channel dependant nonlinearity and for attention.

Implementation of attentive associative memory

The attentive associative memory described in equation [3] can be implemented optically by two vector-matrix multipliers that are connected in a loop with appropriately designed point nonlinearities between them. The schematic diagram of the resulting system is shown in Figure 1. The first vector-matrix multiplication performs an inner product (correlation) between the corrupted input vector and all the stored vectors in parallel. The resultant inner products are transformed via the nonlinear operation F_1 that could be different for each channel. The transformed inner products are then input to the second vector-matrix multiplier, which calculates a linear superposition of the stored vectors. This linear superposition is subject to another nonlinear transform that reflects a priori knowledge about the nature of the data vectors (e.g. positivity, binary values etc.). The result is an improved estimate of the stored vector to which the corrupted input vector corresponds. This estimate is fed back to the first vector-matrix multiplier and the procedure is repeated until a stable state is reached.

The behavior of the attentive associative memory is governed by the nature of the two nonlinearities that operate in the data domain and the inner procuct domain. In this work, binary vectors were chosen as the data vectors. The nonlinear transform chosen

is shown in Figure 2. It contains a threshold level for the input signal below which the output is set to zero, and a saturation level for the output. For intermediate values of the input, the transformation is linear. The critical parameters for this function are threshold and gain values. In our earlier publication[6] we discussed results obtained when these parameters were held fixed during the successive iterations. Although the system still showed the desired behavior of error correction and attention, it was noted that the system can only correct either bit-dropout or cross-talk type of errors at one time. In this paper we describe one method of adaptively setting the values of the parameters during each iteration. The basic principal used is calculating the average level of activity in the data vector or the inner product vector during each iteration and setting the threshold point and the saturation point in the nonlinear transfer function as certain multiples of that value. Thus, the parameters are determined through global activity measurement but are applied to the vectors point-by-point. The result is that values that are below average are suppressed and values that are much above average are clipped to the maximum allowable output value. Since these nonlinearities are incorporated in both the data domain and the inner product domain, the iterative process tends to drive the vector to its nearest neighbor among the set of stored vectors. It should be noted that this is accomplished even when the stored vectors have a high degree of cross-talk. The conventional models of associative memory that are based on correlation matrix formulation either assume orthogonal or near-orthogonal stored vectors, or produce results that are optimum only in a least-squares sense without being exact when the cross-talk is significant. The attention of the system, or the strength of a given stored state, can be manipulated by changing the nonlinear transfer function associated with that vector in the inner product domain. A convenient way of changing the function without changing its shape is to modify the multipliers that determine the threshold and saturation values with respect to the average activity level in the inner product domain. Lowering the threshold as well as the saturation level will increase the attention provided to that vector, and increasing these parameters will reduce the attention given to that vector. This easy manipulation of the strengths of the stored states is not achievable with the conventional associative memory models, as that requires recalculation of all the elements of the correlation matrix in order to accomplish the same purpose.

Computer simulations

The attentive associative memory described in the previous section was simulated on a Personal Computer. Four 16-bit binary vectors were stored in an attentive associative memory. The four vectors to be stored are given below:

A 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0

B 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0

C 1 0 1 0 0 1 0 1 1 0 1 0 0 1 0 1

D 1 0 0 1 0 1 1 0 0 1 1 0 1 0 0 1

It can be seen that these vectors have a large c[illegible]p. This observation can be quantified by calculating inner products between all possible pairs of these vectors. The results are given below:

A * A = B * B = C * C = D * D = 8

A * B = A * C = A * D = B * C = B * D = C * D = 4

The auto- to cross-correlation ratio is therefore only 2:1. In the conventional models of associative memory described by eq. [1], the retrieval works perfectly only when the vectors to be stored are orthogonal to each other. The use of a nonolinearity and an iterative procedure described by Hopfield relaxes the requirement on the set of input vectors to pseudo-orthogonality, but still assumes that the auto- to cross-correlation ratio is equal to square root of N, where N is the size of the vector. Therefore, when the vectors shown in eq. [4] are stored in a conventional associative m[illegible] erroneous results are obtained even when a perfect version of the input vect[illegible] is available for recall. For example, the presentation of vector A leads to a stable state

1 0 1 1 0 0 0 0 1 1 1 0 0 0 0 0

when the vectors are hardclipped at a level 8 to binarize them. A different level for hardclipping changes the nature of the errors but not their number. The use of adaptive nonlinearity of the form described in the earlier section does not help the situation either and will give similarly erroneous results with the data shown in eqaution [4]. This fact indicates that if the stored vectors have a high degree of cross-talk, then using an iterative procedure or a nonlinearity in the data domain is not adequate.

The attentive associative memory contains a provision for introducing a nonlinear transformation on the inner products and hence can suppress cross-talk effectively before the constraints in the data domain are applied to the linear combination of the stored vectors. The parameters for nonlinear transformation in the inner product domain can be determined as follows. When the input vector is the exact version of the stored vector A, the inner products with all of the stored vectors will be (8, 4, 4, 4), thus presenting an average activity level of 5. Since the cross-talk is known to be 4, the threshold can be set at 0.8 times the average activity value. The saturation level can be set at 8, i.e. 1.6 times the average activity value. In the data plane, on the other hand, the average activity value for one of the four stored vectors is 0.5. We can set the threshold of the nonlinear transformation to be the average activity value. Computer simulations were performed with these parameter settings. Even when the input vectors could contain bit dropouts as well as increased cross-talk, the attentive associative memory was shown to extract the stored vector that the input is closest to in the sense of having the largest inner product. We show two examples below from the computer simulations performed.

EXAMPLE I

INPUT VECTOR 1 1 0 1 0 0 0 0 0 1 1 0 0 0 0 0

RETRIEVED VECTOR 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 A

EXAMPLE II

INPUT VECTOR 1 0 1 0 0 0 1 0 1 1 1 0 1 1 0 0

RETRIEVED VECTOR 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 B

In the first example, three bits were missing from the vector A, indicated by the underlined '0'. The attentive associative memory successfully retrieved the full vector A in one iteration. The second example presented a more complicated scenerio. The input vector has an inner product of 6 with vector B and an inner product of 5 with the other three vectors. Thus a combination of cross-talk and bit-dropout reduced the auto- to cross-correlation ratio even further. The attentive associative memory correctly retrieved vector B after three iterations as the stored vector closest to the input vector. Another example was designed to demonstrate the ability of the system to give a higher weight to a chosen vector, thus retrieving it preferentially over the other vectors. The input vector was chosen to have equal value for the inner product with vector A and vector C. The transfer function for channel A was, however, modified by increasing its slope by a factor of 2 compared to the other channels. The attentive associative memory then retrieved A.

EXAMPLE III

INPUT VECTOR 1 0 1 1 0 0 0 1 1 0 0 1 1 1 0 0

RETRIEVED VECTOR 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 A

Without attention, the system can be made to settle into a null state, indicating the input was ambiguous, or it can be made to settle into a partial pattern that is an overlap between vector A and vector C. The particular behavior is determined by the choice for other parameters for the nonlinear transformations.

Optical implementation

Figure 1 indicated that the attentive associative memory contains two vector-matrix multipliers connected in a loop with nonlinearities in between them. The matrix in both the multipliers was, however, identical. This fact can be exploited to design an optical attentive associative memory with bi-directional propogation of light and a common matrix mask. A compact structure can be realized by using long finger-like modulators and detectors to perform the operation of broadcasting and summing, respectively, that are required in vector-matrix multipliers. The schematic diagram of such a compact architecture is shown in Figure 3. The active part of the system consists of an optoelectronic panel containing pairs of detectors and light modulators that are electrically connected to each other through an amplifier and nonlinear circuit. The current out of the detector stripe is proportional to the sum of the light distribution on it. That signal is amplified and processed before applying it to the light modulator stripe, which then broadcasts it to its entire length uniformly. The system shown in Figure 3 is designed to store three vectors, each with four elements. Thus the input

panel contains four pairs of detector-modulator stripes, each three-elements long. The other panel is placed in the correlation domain and contains three pairs of detector-modulator stripes, each four-elements long, that are orthogonally oriented with respect to the input panel stripes. The matrix mask sandwiched between the two panels contains the three four-element vectors. The initial vector can be applied to the input panel via an optical signal to the photodetector or via an electrical signal to the modulator. This vector is then broadcast to all of the vectors of the matrix mask via the stripe modulator. The transmitted light contains the element-by-element multiplication between the initial vector and all the stored vectors. The stripe detector in the correlation domain now sums the products along a row thus performing a vector-vector inner product. These inner product (correlation) results are nonlinearly amplified and applied to the modulators, which broadcast them to the corresponding row vector in the matrix mask. The backward propagating light now performs a scalar-vector multiplication per channel. The detectors in the input pannel now perform a weighted sum of all the stored vectors, thus calculating a new estimate of the initial vector. The detector outputs are nonlinearly amplified before driving the modulator stripes, at which point the cycle repeats.

The system shown in Figure 3 can be simulated with discrete off-the-shelf components, such as LED's and photodetectors. The schematic diagram of the input panel consisting of LED's and photodetectors is shown in Figure 4. Three discrete photodetectors, connected in parallel, replace the stripe detector and three LED's connected in series replace the stripe modulator. An electronic amplifier module per channel implements the desired nonlinear amplification of the signal. An optoelectronic testbed capable of storing four 16-bit vectors was fabricated. Figure 5 shows the photograph of the finished unit. The initial vectors can be input via 16 potentiometers on the front panel. The offset and gain in the correlation domain for each of the four stored vectors can also be controlled via 8 potentiometers on the front panel. One control adjusts the threshold level of the hardclipping operation in the input panel. A film mask was prepared encoding the vectors shown in equation [4]. The operation of this unit was tested and results consistent with the computer simulations were obtained.

Applications of attentive associative networks

The previous sections described a model of an attentive associative network that was used in storing data vectors and retrieving them from incomplete and/or noisy versions of the same vectors. This is only one particular application for the general concept of attentive associative network. The main idea behind the attentive associative network can be described by the following steps:

(1) Project the input vector in the space spanned by the first set of data vectors (possible input vectors)

(2) Transform the projection values [illegible] the adaptive nonlinear transform chosen

(3) Perform a back projection operation on the space spanned by the second set of data vectors (possible output vectors)

(4) Transform the back projected vector using another adaptive nonlinearity that reflects our a priori knowledge about the output domain

(5) Reverse the entire process by first projecting the calculated output vector on the space spanned by the possible output vectors and finally backprojecting on the space of possible input vecotrs

(6) Now the estimate of the input vector can be transformed to reflect our a priori knowledge about the input vector

(7) This transformed estimate of the input vector is [illegible] as the start point for step one and the entire operation is repeated

The result of this iterative process is the calculation of the appropriate output corresponding to the input that is the nearest neighbor of the given vector. Thus, the a priori knowledge constraints in the input as well as the output domain are used in conjunction with the nonlinear suppression of the cross-talk in the projection to carry out the desired mapping between the input and output even when the input is partial and/or noisy. The fundamental operations described above are common to several application domains. In this section, we discuss two particular application areas, namely, Relational Database Architectures and Expert Systems. The aim of this discussion is to provide a plausibility argument for applying the attentive associative network concepts to these fields and encourage the design of appropriate optical architectures to handle these applications.

Relational database architectures

Retrieval of information from databases is a critical activity in a variety of applications. In particular, expert systems require rapid access to their data or knowledge bases. Many expert systems utilize relational databases for storage of knowledge and information. In the following discussion, we describe the general organization of databases and suggest how optical attentive associative memories may be organized to perform as relational database machines.

There are several types of database systems. The most familiar of which are the FILE-MANAGEMENT systems that maintain a file of records. Such systems provide their users with the ability to save and retrieve blocks of information in files. Much like card files, a file-management system allows its user to deal with unit blocks of information. These files suffer from the same drawbacks as files of cards and can only be used to store and retrieve complete records at any time. Additionally, all records are stored and accessible according to one particular index. Advanced file management systems incorporate the additional capability of sorting the records using several different keys or indeces and provide multiple access paths to information. Changing the amount of information that an individual records holds, by adding new fields, usually requires changing all the records of the database. Retrieval of information requires the use of searching, sorting, and selection procedures. A complex query to such a database may require a complex assortment of searches, sorts and selections from retrieved information.

Relational databases resemble file management systems in that they store records of information that are made up of fields. However, information in a relational database may be stored in several different forms or different types of records. Each of these different record types may be thought of as being contained in a separate file. Relational databases allow the user to apply relational algebra operations to these multiple record structures in order to create new structures and files. The operations of Union, Intersection, Projection, Cartesian Product, Set Divison and Set complement can be applied to the database. The power of relational databases is inherited from relational algebra and its capability to implement search and retrieval of information from the database. Any database query can be reconstructed as a series of relational algebra operations.

Relational set operations can be developed from OR, AND and NOT operations. With sufficiently large and sensitive spatial light modulators and detectors, an optical implementation of a relational algebra machine could be constructed and operate at high speeds on a relational database. Union can be performed as an OR of two files. Intersection can be performed as the AND of the NOT of each of two files, the key element in all of these operations will be the ability of the attentive associative memory to implement relations. Figure 6 presents a logical description of a possible mechanism for computing the intersection of Relation A and Relation B. We assume that attentive associtive memories can be used to compute relations and that components capable of computing boolean AND are available. In this system, the entire data set is simultaneously passed through memories A and B to determine Relation A and Relation B and the output is ANDed to determine those points which lie in the intersection of the two relations. In Figure 7, we suggest a mechanism for cascading relations by using the output of one memory as input to another. The first memory acts a filer of selecting objects within its relation, these output objects are reconstituted in an inverted memory and then passed through the memory implementing the second relation.

Associative architectures in expert systems

The ideas developed for applying associative architectures to database systems can potentially impact the performance of expert systems, a well-known discipline within applied artificial intelligence.[8] Expert systems, as the name implies, seek to emulate human expertise in specialized areas, known in AI parlance as domains and embed that within the memory or knowledge base of computing systems. Examples include the interpretation of spectral data (the DENDRAL project), the configuration of minicomputers from components (the R1 project), and the diagnosis of diseases in internal medicine (the MYCIN project). These computer systems "achieve high levels of performance in task areas that, for human beings, require years of special education and training."

Obviously, the amount of knowledge that must be stored in order to be successful in this endeavor is very great, and this places severe demands upon both the organization of memory and the recall processing these systems. Not only do these data bases contain the collection of facts that are relevant to the problem domain of a particular system, but they contain rules that enable the intelligent manipulation of the facts. Two common techniques exist for retrieval: The first uses multiple indices to associatively recall memory elements. The other major type of retrieval is based on pattern matching, where data is retrieved according to some pattern which is related to data categories.

It is important to note that these retrieval schemes differ significantly from those used in numeric computers which store data according to memory addresses. In AI systems, no one knowledge element exactly matches the desired retrieval state, so that very often there are several candidates for recall. A system that can look for the closest match, especially when dealing with incomplete data and references, would reduce delays associated with this process.

In any discussion of retrieval, one must go far beyond the fundamental techniques for recognizing relevant data in the knowledge base to a discussion of how one searches for the set or sets of data that can lead to a defined goal. This is clearly beyond the scope of this paper, but is discussed thoroughly in Ref. 8. However, many expert systems, and most AI systems in general, use the LISP (LISt Processing) language to develop their programs. One feature of this language is its ability to process information recursively, and hence generate an expectation for an element to be retrieved from memory. This expectation, which is derived from the state of the machine at the specific point in time, is clearly analogous to the concept of "attention" in the associative architecture discussed so far.

Finally, it is the combination of the searching of the knowledge base and the retrieval techniques discussed above that form the major computational bottleneck in expert systems. The searching over extensive knowledge bases is typically reduced by using knowledge about the systems data base to "prune" the search process. However, for each element searched, an association or matching operation must be performed, causing the problem to worsen as knowledge base size scales. Memory organizations which can effectively delimit the size of the search space and can efficiently match patterns, such as the attentive associative memory, should help alleviate this bottleneck.

References:

1. Grossberg, S., "Studies of Mind and Brain," D. Reidel, Boston, 1981.

2. Kohonen, T., "Self Organization and Associative Memory," Springer-Verlag, New York, 1983.

3. Hopfield, J. J., "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," Proceedings of the National Academy of Sciences, USA 79, p. 2554, 1982.

4. Psaltis, D., Farhat, N., "Optical Information Processing Models Based on Associative Memory Models of Neural Networks with Thresholding and Feedback," Opt. Lett. Vol. 10, p. 98, 1985.

5. Fisher, A. D., Giles, C. L., Lee, J. N., "Adaptive Associative Memory Models," Presented at Optical Computing Topical Meeting, March 18-20, 1985, Lake Tahoe, Nevada.

6. Athale, R. A., Szu, H. H., Friedlander, C. B., "Attentive Associative Memory and Its Optical Implementation," Submitted to Optics Letters and Presented at The Annual Meeting of Optical Society of America, Washington, DC, 1985.

7. Soffer, B. H., Dunning, G. J., Owechko, Y. Marom, E., "Associative Holographic Memory with Feedback Using Phase Conjugate Mirrors," Submitted to Optics Letters and Presented at The Annual Meeting of Optical Society of America, Washington, DC, 1985.

8. Hays-Roth, F., Waterman, D. A., and Lenat, D. B., "Building Expert Systems," Addison-Wesley Publication, 1983.

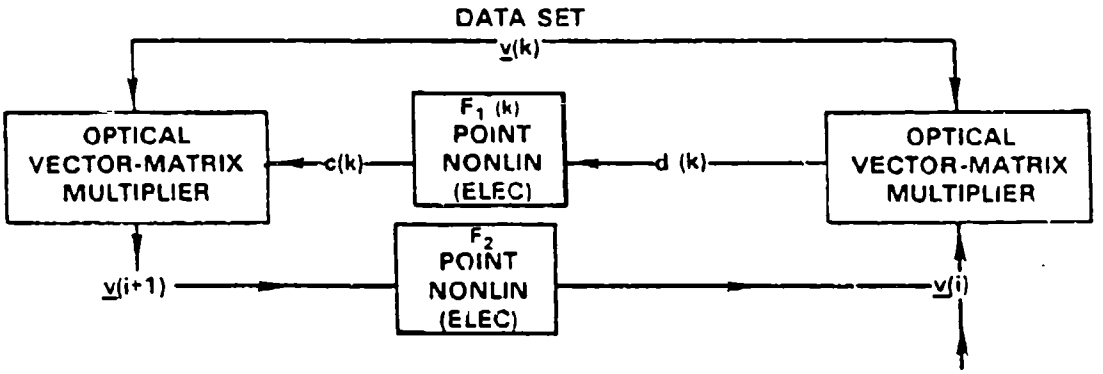


Figure 1. The block diagram of an optical attentive associative memory.

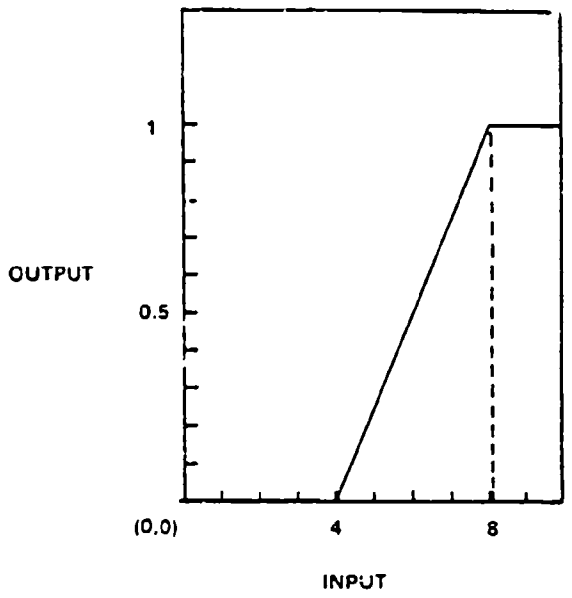


Figure 2. The nonlinear transfer function applied to the inner products (corresponds to F1 (k) in Figure 1).

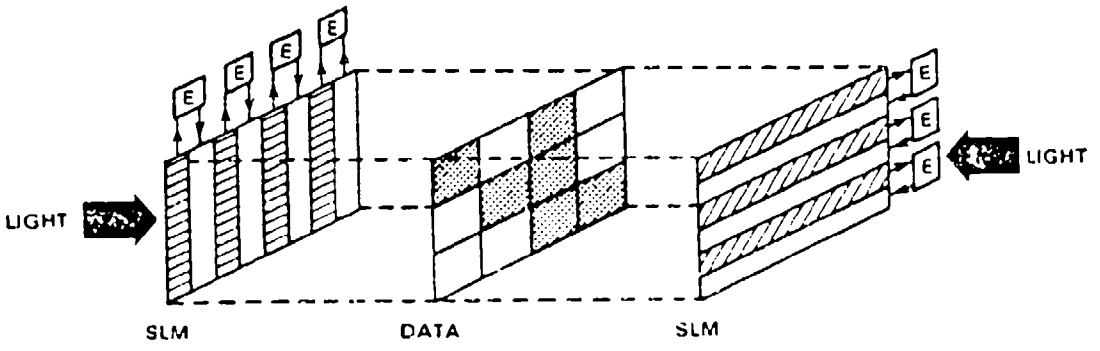


Figure 3. A compact optical system implementation an optical attentive associative memory. The cross-hatched stripes correspond to detectors, the open stripes correspond to modulators, and the boxes labeled 'E' represent an electronic nonlinear amplifier circuit. The data is encoded on a film mask or a real-time mask.

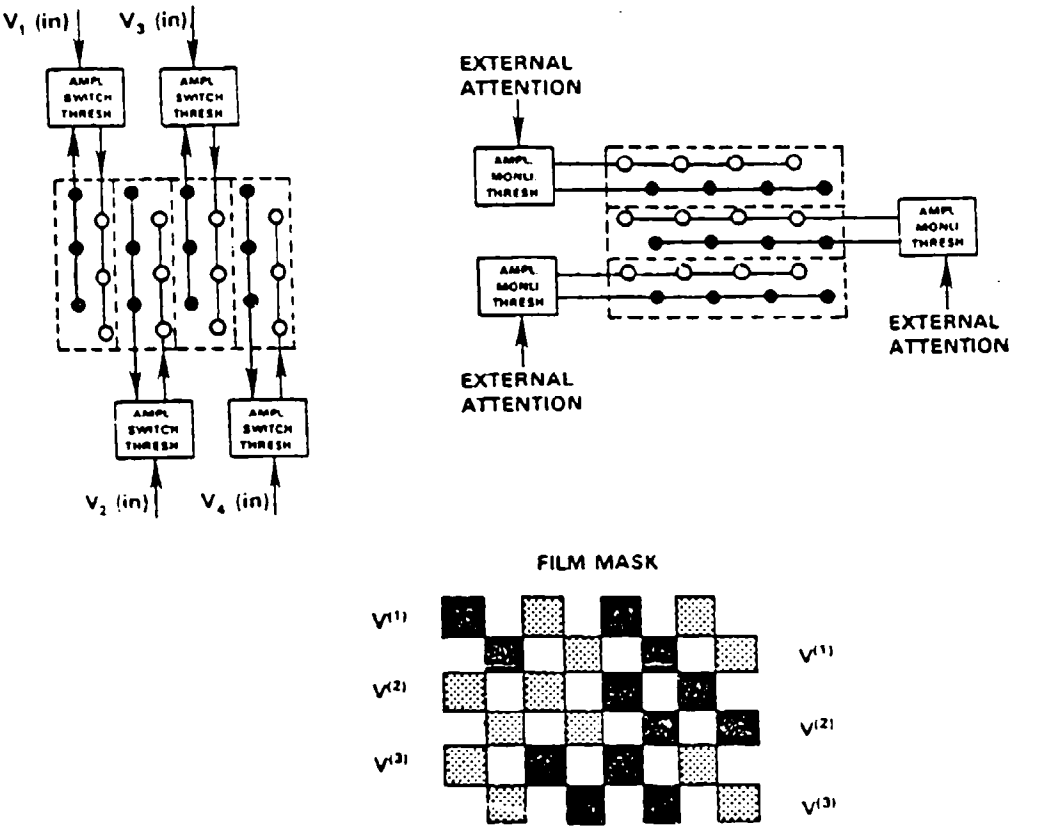


Figure 4. The schematic diagram of the components of an optoelectronic testbed simulating the design shown in Figure 3 with discrete components.

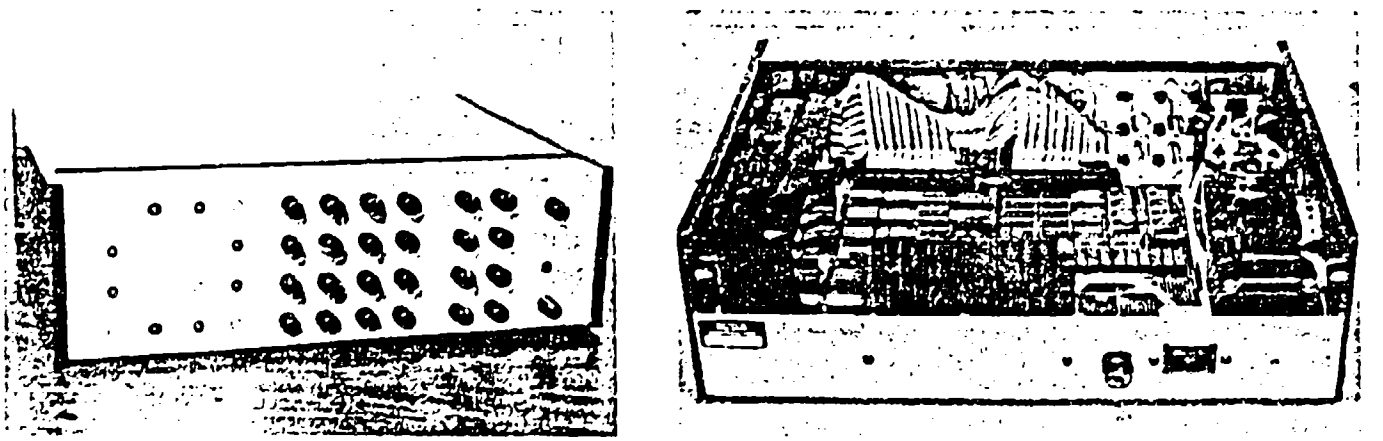


Figure 5. The photographs of an optoelectronic testbed demonstrating an optical attentive associative memory.

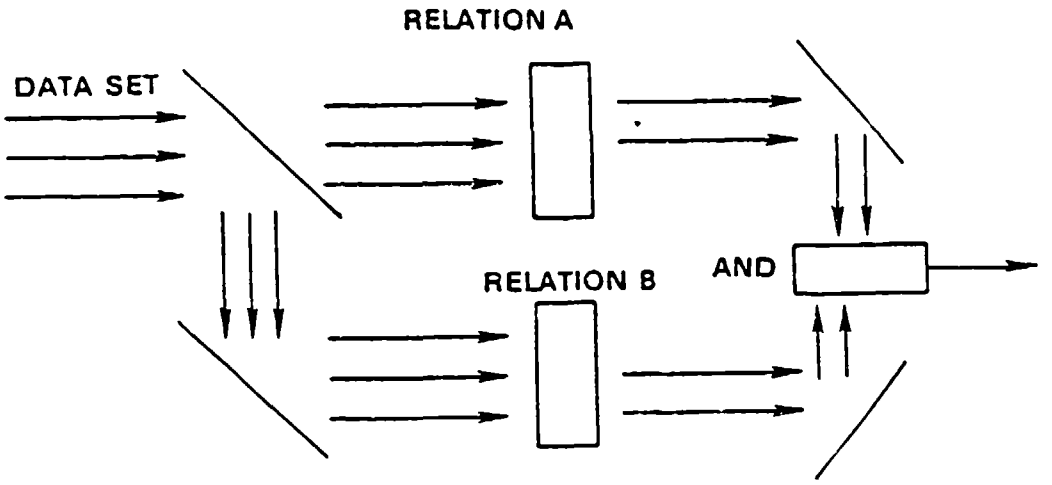


Figure 6. Schematic diagram of a system for computing an intersection between relation A and relation B.

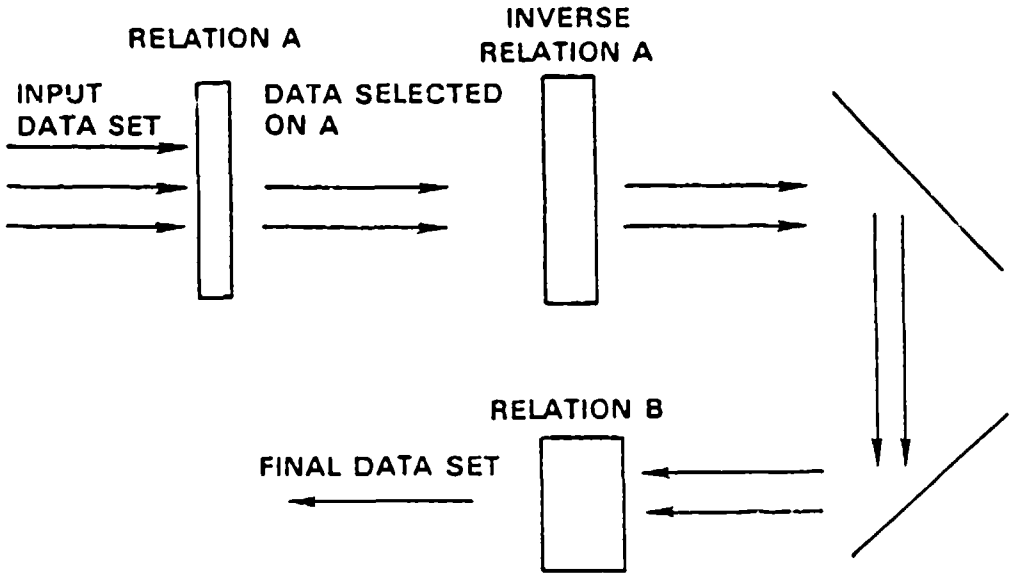


Figure 7. Schematic diagram of a system for cascading relations A and B.

OPTICAL MATRIX PROCESSORS

ARCHITECTURES FOR OPTICAL MATRIX MULTIPLIERS

Ravindra A.Athale

The BDM Corp.
7915 Jones Branch Drive
McLean, VA.22102

ABSTRACT

The operation of multiplying two matrices is one of the fundamental operations that is frequently encountered in signal processing, image processing and numerical computations. This operation can be performed by optical systems employing parallelism. This paper discusses different architectures for optical processors performing this operation. The architectures are divided into several classes according to the degree of parallelism entailed and the type of interconnections employed.

INTRODUCTION

The operation of multiplying two matrices is one of the most common operations encountered in signal processing, image processing, and numerical computation involving solutions of a system of linear simultaneous equations. This operation is mathematically defined in equation 1 below:

$$C_{ij} = \sum_{k} A_{ik} B_{kj} \qquad [1]$$

For i=1 to N and j=1 to N

In FORTRTAN , this operation can be coded by the following program:

```
      DIMENSION   A(N,N),   B(N,N),   C(N,N)
      DO 10 I=1,N
      DO 10 J=1,N
      DO 10 K=1,N
      C(I,J)  = C(I,J)  + A(I,K)  *  B(K,J)
10    CONTINUE
```

It should be noted that there are three DO loops in the program with N passes per loop. Since the operations performed in the loops are multiplication and addition, this program involves N^3 multiplications and additions. The indices I and J orrespond to the array index of the output matrix C, and the index K is the common array index for the input matrices A and B, over which summation is performed

Optical processing systems are capable of performing the operations of analog multiplication and addition in parallel between one- or two-dimensional array of positive real numbers and achieve global communication between them. This property of optics has been used in the past for signal processing and image processing operations primarily based on Fourier transforms. Recently, optical processing systems have been investigated for performing matrix operations in parallel, which will make them more widely applicable. The large variety of optical architectures reported in the literature can be classified into three different categories according to the level of parallelism:

(i) one-dimensional systems performing N operations in parallel
(ii) two-dimensional systems performing N^2 operations in parallel
(iii) two-dimensional systems employing multiplexing performing N^3 operations in parallel.

The first two categories can be further subdivided according to which of the DO loops in the program for matrix multiplication are implemented in parallel. The optical processors in each subcategory can be implemented with different technologies, such as integrated optics, acoustooptics, electrooptics etc. A further classification results when one considers which parameters - e.g. time or space, temporal or spatial frequency - are used to multiplex the operations and which are used for summation/integration. A truely comprehensive study which includes a detailed analysis of all of these variations will indeed be massive. In this paper we will briefly discuss the different optical architectures in a generic way. We will place particular emphasis on the types interconnects involved in each of these archietctures since it will high light the special advantages offered by the use of optics.

Other details can be found in the articles that are referenced at the end of the paper.

N-PARALLEL OPTICAL SYSTEMS:

The optical processor in this category perform N multiplications and/or additions in parallel. If we look at the program listing, we will get three choices for the DO loop index that we could implement in parallel with an optical system. The indices I and J correspond to the array indices of the output matrix C, and hence are equivalent for the purpose of this analysis. Hence there are two choices, (i)parallelizing the DO loop over index I (or J), and (ii)parallelizing the DO loop over index K.

(i) This choice of the DO loop index (I) leads to optically performing N multiplication in parallel and the summation over the K index and the DO loop over J sequentially. The resulting optical architecture is shown in Figure 1. This system contains a one-dimensional (1-D) spatial light modulator (SLM), a point modulator, a 1-D time-integrating detector array, and collimating optics (for signal broadcasting) and imaging optics. The basic operation that this processor performs in one clock cycle of the 1-D SLM is that of scalar-vector multiplication defined by equation 2:

$$c = a\,\underline{b} \qquad [2]$$

It should be noted that this system uses an interconnection network for broadcasting in one dimension.

(ii) This choice of the DO loop index (K) leads to performin N multiplications and additions in parallel. The resulting architecture is shown in Figure 2. This system contains two 1-D SLM, a point detector, and focussing optics (for fan-in) and imaging optics. The basic operation that this processor performs in one clock cycle of the 1-D SLM is that of vector inner product defined by equation 3:

$$c = \underline{a}^T\,\underline{b} \qquad [3]$$

It should be noted that this system uses an interconnection network for fanning in N-data channels in one dimension.

Both of these optical architectures do not fully exploit the 2-D parallelism of optics. However, their 1-D nature allows for an integrated optic implementation that allows for higher speed operation (100 MHz) in a small and rugged package. It should also be noted that the operations described above are of interest in and by themselves for signal processing/symbolic processing operations and do not have to be considered in the context of matrix multiplications alone. The scalar-vector multiplication is is useful in calculating a weighted version of a given input and the vector-vector inner product can give a similarity measure between the two vectors to be compared.

N^2-PARALLEL OPTICAL SYSTEMS

The optical processors in this category perform N^2 multiplications and/or additions in parallel and hence fully exploit the 2-D parallelism offered by optics. If we look at the program listing, we will get two distinct choices for the two DO loops that we can implement in parallel with an optical system. (i) this choice involves performing the DO loops over I and J index in parallel, (ii) the second choice involves performing the DO loops over I (J) and K in parallel.

(i) This choice of DO loop indices (I and J) leads to performing N^2 multiplications in parallel and the summation over the K index sequentially. The resulting optical architecture is shown in Figure 3. This system contains two 1-D SLMs arranged orthogonal to each other, optics for collimating and focussing simultaneously along orthogonal directions, and a 2-D time-integrating detector array. The basic operation that this processor performs in one clock cycle of the 1-D SLM is that of a vector-vector outer product defined in equation [4]:

$$C' = \underline{a}\,\underline{b}^T \qquad [4]$$

where C' is a rank one matrix that is NXN in dimension. It should be noted that this optical architecture uses two 1-D input arrays and yet calculates a 2-D output array. The interconnections utilized by this system are quite complex in that they involve broadcasting along one spatial dimension and fanning-in along orthogonal spatial direction. One element of the 1-D SLM encoding one element of the vector b is simultaneously accessed by N elements of the other input vector a without contention. The data path for an element of vector a retains

its identity. So after being multiplied by an element of vector $\underline{b}$, it is directed to a specific location on the 2-D time-integrating detector array as shown in Figure 3. This system can be considered to be a spatially multiplexed version of the 1-D scalar-vector multiplier depicted in Figure 1.

(ii) This choice of DO loop indices (I and K) leads to performing N^2 multiplications and additions in parallel and performing the DO loop over the remaining index (J) sequentially. The resulting optical architecture is shown in Figure 4. This system contains one 1-D SLM, one 2-D SLM, one 1-D non-integrating detector array, and optics for imaging/collimating on the input side and for imaging/focussing on the output side. The basic operation that this processor performs is that multiplying a column vector of matrix B by matrix A defined in equation [5]:

$$\underline{c} = A\,\underline{b} \qquad [5]$$

The interconnections utilized by this system involve broadcasting the vector $\underline{b}$ to all rows of matrix A on the input side and fanning in the product of a row of A and vector $\underline{b}$ on to a specific detector element on the output side. This system can be considered to be a spatially multiplexed version of the 1-D vector-vector inner product processor depicted in Figure 2.

The optical processors in this category do utilize the 2-D parallelism of optical systems by employing the spatial variables for multiplexing and/or for integration. These two operations are also useful in and by themselves in signal and image processing. The operation of outer product is critical in synthesizing a complicated matrix (e.g. an image) from several simple "primitive" images corresponding to rank one matrices. The vector-matrix multiplication implements a generalized linear transformation on a 1-D input. The optics utilized by these systems contains off-the-shelf components like spherical and cylindrical lenses to give different properties along the two orthogonal directions.

N^3-PARALLEL OPTICAL SYSTEM:

The optical processors in this category perform N^3 multiplications and additions in parallel. Since the matrix-matrix multiplication involves N^3 multiplications/additions this class of optical processors will perform the operation in one clock cycle of the active devices involved. Since we are performing all the DO loops in parallel, there is only one way of formulating the processor mathematically. Different architectures result, however, when designing such a system optically.

The schematic diagram of the N^3-parallel optical system is shown in Figure 5. This system uses two 2-D SLMs for inputing matrices A and B and a 2-D detector arrayfor detecting the output matrix C. The optics involved is complicated and has to be implemented via computer generated holography.

One intriguing feature of this architecture is that it employs only N^2 active elements and still performs N^3 operations in parallel. Therefore the efficiency of this architecture as a parallel processor (computational speedup / number of processing elements) is N which can be much larger than 1! In most other designs for parallel processors the goal is to achieve an efficiency of 1, indicating a linear speedup with the number of processors. This indeed represents a unique feature of optics in that it utilizes the parallel, contention free, multiple-access communication capability of optics to add an extra dimension to the computational power of a parallel processor. Since the operation of matrix-matrix multiplication is a well structured operation with little interdependance between the calculations of the elements of the output matrix, this feature of optical systems can be exploited to fullest extent to achieve superlinear speedup in a parallel architecture.

The N^3-parallel optical processor can be viewed as a multiplexed version of the N^2-parallel optical architectures. Since in the earlier section we discussed two different optical systems for those architectures, we can view the N^3-parallel architecture as multiplexed versions of either the outer product optical processor or the vector-matrix optical processor. It should be emphasized that this division is strictly for conceptual clarity and is not indicative of a deeper category.

Figure 6(a) shows the schematic diagram of an N^3-parallel optical processor viewed as N outer product optical processors that are spatially

multiplexed. In the system depicted in Figure 3 the addition of the outer product matrices for calculating the output matrix C was performed by integration in time on the 2-D detector array. In Figure 6(a), all the outer products are simultaneously calculated and are therefore added by integration of N terms in space by fanning in the apropriate signals emerging from the elements of the second matrix B. Each column of matrix A is encoded by a light beam traveling at an apropriate angle so as to interact with the correct row of matrix B as indicated in Figure 3. On the output detector array, the results of the different outer products performed in parallel converge thus performing the final integration. Figure 6(a) shows the optical paths for only two outer products for clarity.

Figure 6(b) shows the schematic diagram of the N^3-parallel optical processor viewed as a spatially multiplexed version of the optical vector-matrix multiplier shown in Figure 4. In that system, the operation of vector-matrix multiplication was performed in one cycle generating one row of the output matrix C. The full answer was calculated by generating different rows of matrix C in a time-sequential fashion. In the system depicted in Figure 6(b), all rows of matrix A are available simultaneously while the matrix B is used in N different vector-matrix products simultaneously. The resultant N rows of the output matrix C are spatially separated and are detected by the rows of the output detector array. In this processor, each row of matrix A is encoded by a light beam traveling at an appropriate angle. At matrix B, all rows of A are simultaneously accesing the elements of B while keeping their distinct identity. The unique encoding of the light beam for each row of A causes the output to be separated on the detector array thus providing all rows of matrix C in parallel.

CONCLUSION

Optical processors offer the unique features of two-dimensional parallelism in the basic arithmetic operations of analog multiplications and additions and global, parallel, and contention-free communication between the two-dimensional array of simple processing elements. These features can be exploited to build optical processors for performing the general operation of matrix-matrix multiplication. In this paper, we outlined several different architectures of parallel optical processors for performing this operation with varying degree of parallelism. In each category the type of communication involved were emphasized. A unique optical architecture was described that uses the contention-free communication offered by optics to obtain a speedup of N^3 with a system containing only N^2 active processing elements.

REFERENCES

GENERAL OPTICAL PROCESSING REFERNCES:

J.W.Goodman, "Introduction to Fourier Optics", McGraw-Hill, 1968.

S.H.Lee ed. "Optical Information Processing: Fundamentals", New York: Springer-Verlag, 1981.

Proceedings of IEEE Special Issue on Optical Computing, Volume 65, No.1, January 1977.

Proceedings of IEEE Special Issue on Optical Computing, Volume 72, 1984.

GENERAL REVIEW ARTICLES ON OPTICAL MATRIX PROCESSORS:

R.A.Athale, 10th International Optical Computing Conference, April 6-8, 1983, Cambridge, Mass. IEEE Catalog Number 83CH1880-4.

D.Casasent, Proc. IEEE, Vol.72, p.831,1984.

W.T.Rhodes and P.S.Guilfoyle, Proc. IEEE, Vol.72, p.820, 1984.

1-D INTEGRATED OPTICAL ARCHITECTURES:

C.Verber, Proc. IEEE, Vol.72, p.942, 1984.

OPTICAL VECTOR-MATRIX MULTIPLIERS:

M.A.Monahan, K.Bromley, and R.P.Bocker, Proceedings of IEEE, Vol.65, p.121,. 1977.

J.W.Goodman, A.R.Dias, and L.M.Woody, Optics Letters, Vol.2,p.1, 1978

H.J.Caulfield, W.T.Rhodes, M.J.Foster and S.Horwitz, Optics Comm., Vol.40, p.80, 1981

OPTICAL OUTER PRODUCT PROCESSORS:

R.A.Athale and W.C.Collins, Applied Optics, Vol.21, p.2088, 1982.

R.A.Athale and J.N.Lee, Proc. IEEE, Vol.72, p.931, 1984.

R.P.Bocker, H.J.Caulfield, and K.Bromley, Applied Optics, Vol.22, p.804, 1983.

OPTICAL FULLY PARALLEL PARALLEL MATRIX-MATRIX MULTIPLIERS:

R.A.Heinz, J.O.Artman, and S.H.Lee, Applied Optics, Vol.9, p.2161, 1970.

P.N.Tamura and J.C.Wyant, Proceedings of SPIE, Vol.83, 1975.

A.R.Dias, in "Optical Information Processing for Aerospace Applications", NASA Conference Proceedings 2207, (NTIS, Springfield, VA.).

Y-Z Liang and H.K.Liu, Optics Letters, Vol.9, p.322, 1984.

H.Nakano and K.Hotate, Applied Optics, Vol.24, p.4238, 1985.

OPTICAL MATRIX PROCESSORS: ALGORITHMS AND HIGHER ORDER OPERATIONS

H.J.Caulfield, D.Dvore, J.W.Goodman, and W.T.Rhodes, Applied Optics, Vol.20, p.2263, 1981.

D.Casasent and M.Carlotto, Applied Optics, Vol.21, p.147, 1982.

P.A.Athale and J.N.Lee, Optics Letters, Vol.8, p.590, 1983.

D.Casasent, C.Neuman, and J.Lycas, Applied Optics, Vol.23, p.1960, 1984.

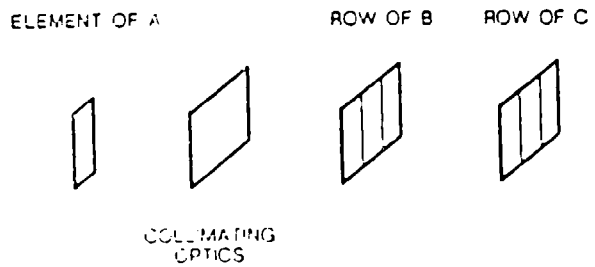


Figure 1 The N-parallel scalar-vector product optical processor. (Imaging optics ommitted.)

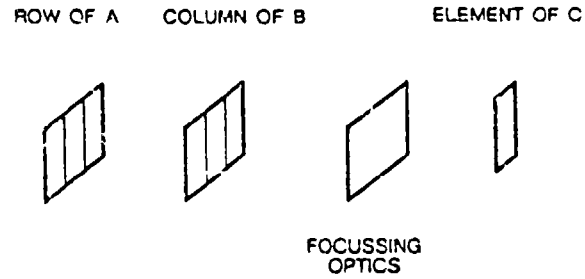


Figure 2. The N-parallel inner product optical processor. (Imaging optics ommitted.)

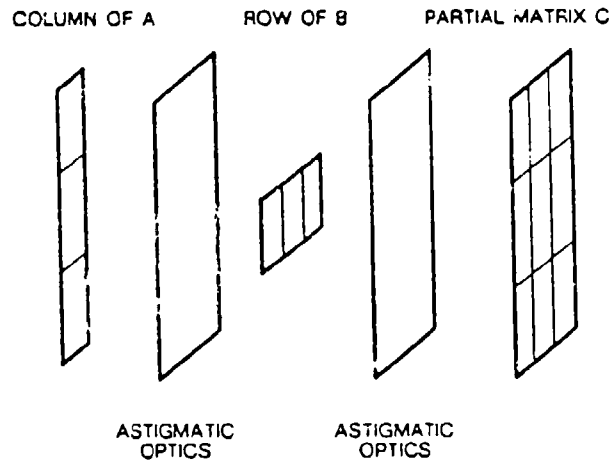


Figure 3. The N^2-parallel outer product optical processor.

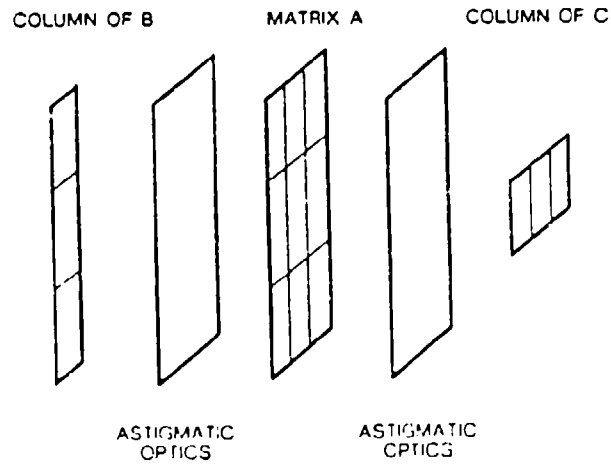


Figure 4. The N^2-parallel vector-matrix optical processor

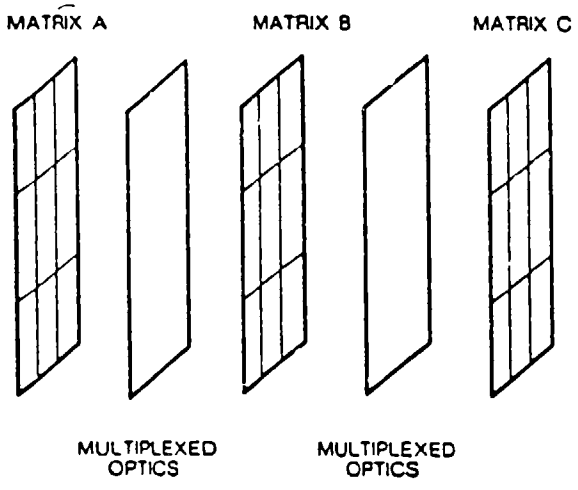


Figure 5. The N^3-parallel optical processor.

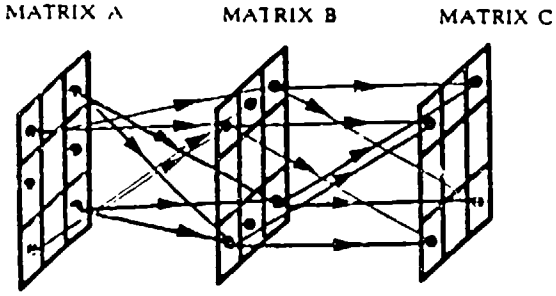


Figure 6(a). The N^3-parallel optical processor viewed as a multiplexed outer product processor.

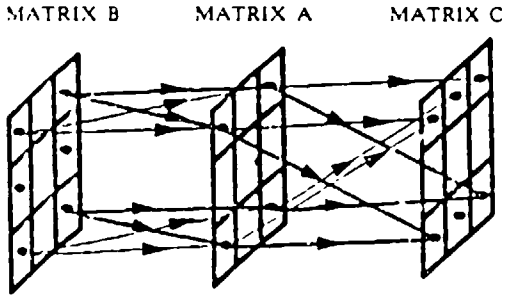


Figure 6(b). The N^3-parallel optical processor viewed as a multiplexed vector-matrix processor.